

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT SECURITY CLASSIFICATION

AD-A211 115

30.2.1989

1b. RESTRICTIVE MARKINGS

None

DISTRIBUTION/AVAILABILITY OF REPORT

3. MONITORING ORGANIZATION REPORT NUMBER(S)

ARU 23439.6-EG

6a. NAME OF PERFORMING ORGANIZATION

The Trustees of Columbia University
in the City of New York6b. OFFICE SYMBOL
(If applicable)

7a. NAME OF MONITORING ORGANIZATION

U.S. Army Research Office

6c. ADDRESS (City, State, and ZIP Code)

Office of Projects and Grants
Box 20, Low Memorial Library
New York, New York 10027

7b. ADDRESS (City, State, and ZIP Code)

P.O. 12211
Research Triangle Park, NC 27709-22118a. NAME OF FUNDING/SPONSORING
ORGANIZATION

Army Research Office

8b. OFFICE SYMBOL
(If applicable)

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

DAAL03-86-K-0071

8c. ADDRESS (City, State, and ZIP Code)

PO Box 12211
Research Triangle Park, NC 27709-2211

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

The Development of an Expert System for The Creative Design of Mechanisms

12. PERSONAL AUTHOR(S)

Ferdinand Freudenstein and David A. Hoeltzel

13a. TYPE OF REPORT

13b. TIME COVERED

FROM 5-1-86 TO 4-30-89

14. DATE OF REPORT (Year, Month, Day)

6-26-89

15. PAGE COUNT

111

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

ARTIFICIAL INTELLIGENCE, POLYNOMIALS, ALSO T.M.S.
COMPUTER AIDED DESIGN

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The Automatic Identification, Assemblability and Sketching of Mechanisms

A unified design methodology has been developed for the creative design of mechanisms. This is based on an abstract representation of the kinematic structure of mechanisms, which we have called the stratified representation of mechanisms. This transforms the basic structure of a mechanism into an essentially unique spanning tree with a minimum number of edges, each associated with a fundamental loop. In this fashion hundreds of thousands of mechanisms have been generated automatically, thus providing a creative design tool of extraordinary power and efficiency.

The same basic approach has been developed for the automatic determination of favorable mechanism dimensions by means of algorithms for the determination of the rotatability and assemblability of the links, while eliminating crossing links. *Kay words*

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT

☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION

22a. NAME OF RESPONSIBLE INDIVIDUAL

Dr. Gary Anderson

22b. TELEPHONE (include Area Code)

22c. OFFICE SYMBOL

DD Form 1475, JUN 86

Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

DISTRIBUTION STATEMENT A

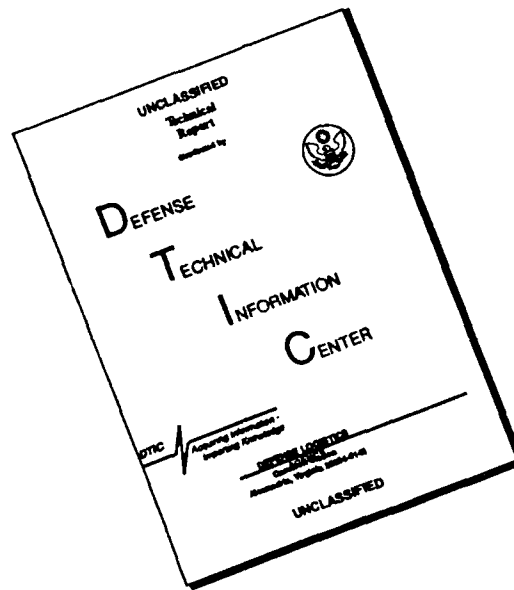
Approved for public release;
Distribution Unlimited

89

7 31 120

Unclassified

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

Final Technical Report on
The Development of an Expert System for the
Creative Design of Mechanisms

Submitted to
The United States Army Research Office

by
Ferdinand Freudenstein
Higgins Professor of Mechanical Engineering

and
David A. Hoeltzel
Associate Professor
Department of Mechanical Engineering
Columbia University
New York, New York 10027-6699

June 15, 1989

Supported Under U.S. Army Research Office Grant No. DAAL03-86-K0071

Preface

The contents of this technical report constitute a detailed summary of the research performed under grant DAAL03-86-K0071, supported through the United States Army Research Office. The work has been subdivided in a logical manner, and includes an introductory discussion of the importance of conceptual design and its impact on CAD. Subsequent chapters discuss knowledge-based approaches to mechanism synthesis, kinematic analysis, redesign, design optimization and a new "stratified code" methodology for the efficient representation of kinematic structure.

More specifically, this report is focused on an expert system approach to mechanism design, the contents of which embodies five parts. Part I includes chapters 1 and 2, and introduces conceptual design in mechanical engineering. Part II includes chapters 3, 4 and 5, and provides some insight into mechanism design synthesis. Part III includes chapter 6, and demonstrates kinematic analysis, through a new approach based on object-oriented programming. Part IV includes chapters 7 and 8, and provides a description of several aspects and applications of nonlinear programming as it applies to mechanism design optimization. Part V presents a innovative hierarchical methodology for representing the kinematic structures of mechanisms, referred to as the "stratified code" representation.



Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By <i>per Form 50</i>	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
<i>A-1</i>	

Selected Publications

1. "The Stratified Representation of Mechanisms", by W.E. Fang and F. Freudenstein, Trends and Developments in Mechanisms, Machines and Robotics, ASME Publication DE-Vol 15-1, 1988, pp. 115-123.
2. "A Hierarchical Methodology for the Creative Design of Mechanisms", by W.E. Fang, Doctoral Dissertation, Department of Mechanical Engineering, Columbia University, May, 1989, (Available from University Microfilms, Ann Arbor, Michigan).
3. "A Combinatorial Approach to the Automatic Sketching of Planar Kinematic Chains and Epicyclic Gear Trains", by W. H. Chieng and D. A. Hoeltzel, Trends and Developments in Mechanisms, Machines and Robotics, ASME Publication DE-Vol 15-1, 1988, pp. 79-90.
4. "Knowledge Representation and Planning Control in an Expert System for the Creative Design of Mechanisms", by D.A. Hoeltzel, W.H. Chieng and J. Zissimides, J. of Artificial Intelligence in Engineering Design Analysis and Manufacturing, Academic Press, Vol. 1, No. 2, 1988, pp. 119-137.
5. "Conceptual Design Tools for the Creative Synthesis, Analysis and Redesign of Mechanisms - MECXPRT {Mechanism Expert}", by W.H. Chieng, Doctoral Dissertation, Department of Mechanical Engineering, Columbia University, August, 1989.

Table of Contents

	<u>Pages</u>
Part I: Introduction	
1. Introduction to Conceptual Design	4 - 7
2. Relationship Between Conceptual Mechanism Design and Artificial Intelligence	8 - 13
Part II: Mechanism Design Synthesis	
3. Tree Search Algorithms for the Enumeration of Design Concepts	14 - 22
4. Knowledge Representation and Planning Control for Mechanism Design	23 - 31
5. Clustering Analysis in Creative Mechanism Design	32 - 42
Part III: Mechanism Design Analysis	
6. Kinematic Analysis Based on Object-Oriented Programming	43 - 52
Part IV: Mechanism Redesign and Design Optimization	
7. Concepts of Numerical Optimization - Application: Mechanical Error Analysis	53 - 64
8. Hybrid (Symbolic - Numeric) Optimization Methodology	65 - 75
Part V: A Hierarchical Methodology-The Stratified Code	
	80 - 111

All **bold**, *italicized* words and phrases are defined in this section.

- Adjacency matrix:

An adjacency matrix is a square and symmetric matrix that is used to store the connectivity information for the links of a mechanism. For example, if the element in the i th row and j th column of the adjacency matrix is equal to 1, i.e. $M(i,j) = 1$, this implies that the i th vertex (link) is connected to the j th vertex (link). If $M(i,j) = 0$, this implies that the i th vertex (link) is not connected to the j th vertex (link).

- Artificial intelligence:

The study of techniques for solving exponentially hard problems in polynomial time, by exploiting knowledge about the problem domain [Rich, '83].

- Backward matching:

Represents matching from a desired goal, backwards through the steps in the solution which achieve that goal.

- Computational efficiency:

The computational efficiency of an algorithm is judged by a complexity measurement for the given algorithm. The complexity is typically expressed in terms of the number of variables which are involved. For example, if an algorithm has only one variable, denoted by n , and if required cpu time for different values of n is kn^3 , where k is a constant coefficient, this algorithm is referred to as a polynomial time algorithm of order 3, or $O(n^3)$, using big O notation.

- Computational sufficiency:

The computational sufficiency of an algorithm represents its ability to handle degenerate cases. For a computer-based numerical calculation, the computational sufficiency is measured by the correctness of the result in terms of the magnitude of the computational truncation error. For a computer-based symbolic manipulation, the computational sufficiency is measured in terms of the invertability of the symbols. For example, the given algorithm must be able to handle $[A]^{-1}$ when $[A]$ is of degenerate rank.

- Heuristic:

A technique that improves the efficiency of a search process, possibly by sacrificing claims of completeness [Rich, '83].

- Hybrid (symbolic - numeric) design system:

A hybrid design system explicitly separates a design process into a symbolic portion and a numerical design portion. This coupled approach to design usually starts with a symbolic layout of the design and proceeds to numerical optimization of the design in accordance with the symbolic layout.

- Integrated design decision making:

An integrated approach to obtaining a solution to a design problem.

- Isomorphism:

A graph mapping from $G \rightarrow G'$ is called an "isomorphism" if both its vertex function and its edge function are one-to-one and onto. Two graphs

are called isomorphic if there exists an isomorphism from one to the other. [Tucker, '84].

3

- Operators (in design planning):

Data elements which contain functions which permit the representation and control of knowledge.

- Production memory (rule based expert system terminology):

Production memory stores the production rules which are usually in form of If-Then statements.

- Recognize-selection-Act Cycle (rule based expert system terminology):

There are three processes which must be cycled through within an inference engine 1) Recognition: the inference engine applies all the production rules to the working memory elements to check if the condition part (RHS) of any rule is satisfied 2) Selection: when there is more than one rule whose condition part is satisfied, the inference engine will judge which rule is more important than the others, according to certain predefined selection criteria, and 3) Act: the inference engine will either produce or delete the work memory elements or execute the external functions in accordance with the action part (LHS) of the selected rule.

- Rule matching (rule based expert system terminology):

Rule matching is the recognition capability of the inference engine. Rules in the production memory are matched with the elements in the working memory. This process includes 1) value to value matching, 2) value to variable binding, and 3) variable to variable instantiation.

- State variable (in design planning):

State variables are indices which implicitly show the state of the design process. State variables are used to control the design planning process.

- Subdesign decisions:

These are decisions to be made during the solution of a design problem which occur as nodes in a search tree. Since typically several, and oftentimes many, of these must be made in order to effect a successful solution, they are referred to as subdecisions.

- Transparency between objects (in a knowledge base):

When inheritance and relations for hierarchically structured data is unlimited, i.e., children nodes inherit all property values from their father node and the grandchildren nodes inherit the property values from the children nodes, this is referred to as fully transparent inheritancy. When inheritance and relations for hierarchically structured data is conditional i.e., the children nodes do not necessarily inherit all property values from their father node or grandfather node, this is referred to as semi-transparent inheritancy. In semi-transparent inheritancy, the conditions which allow inheritancy and relation propagation must be additionally specified.

- Working memory (rule based expert system terminology):

Working memory stores all the data elements which will be used to match and fire the production rules.

1. Introduction to Conceptual Design

A number of researchers [Atkinson, '85] and [Vollbracht, '86] have shown that the features, performance, cost, and reliability of a new product are, to a large extent, determined during the conceptual design stage of the overall design process. A diagram which shows the distribution of dollar cost and commitment over the design cycle of a product is shown in Figure 1.1. Consequently, the elaboration and evaluation of processes germane to conceptual design has been given serious consideration in recent years.

The conceptual phase of mechanical design can be separated into three levels that include a) a functional level, b) a component connectivity level, and c) a component level [Chieng and Hoeltzel, '89]. As shown in Figure 1.2, the conceptual design of a hard disk drive can be expressed in the form of a block diagram at the functional level. This describes the input and output control features required for a hard disk drive. Furthermore, the hard disk drive can be described from the viewpoint of component connectivity. Based on a connectivity diagram, the relationships between adjacent components can be first established, and their influence on one another can then be analyzed. The concept of component level design contains two sets of information, one is information about geometric attributes, and the other is information about machining attributes.

This section of the thesis proposal discusses only the conceptual phase of mechanism design.

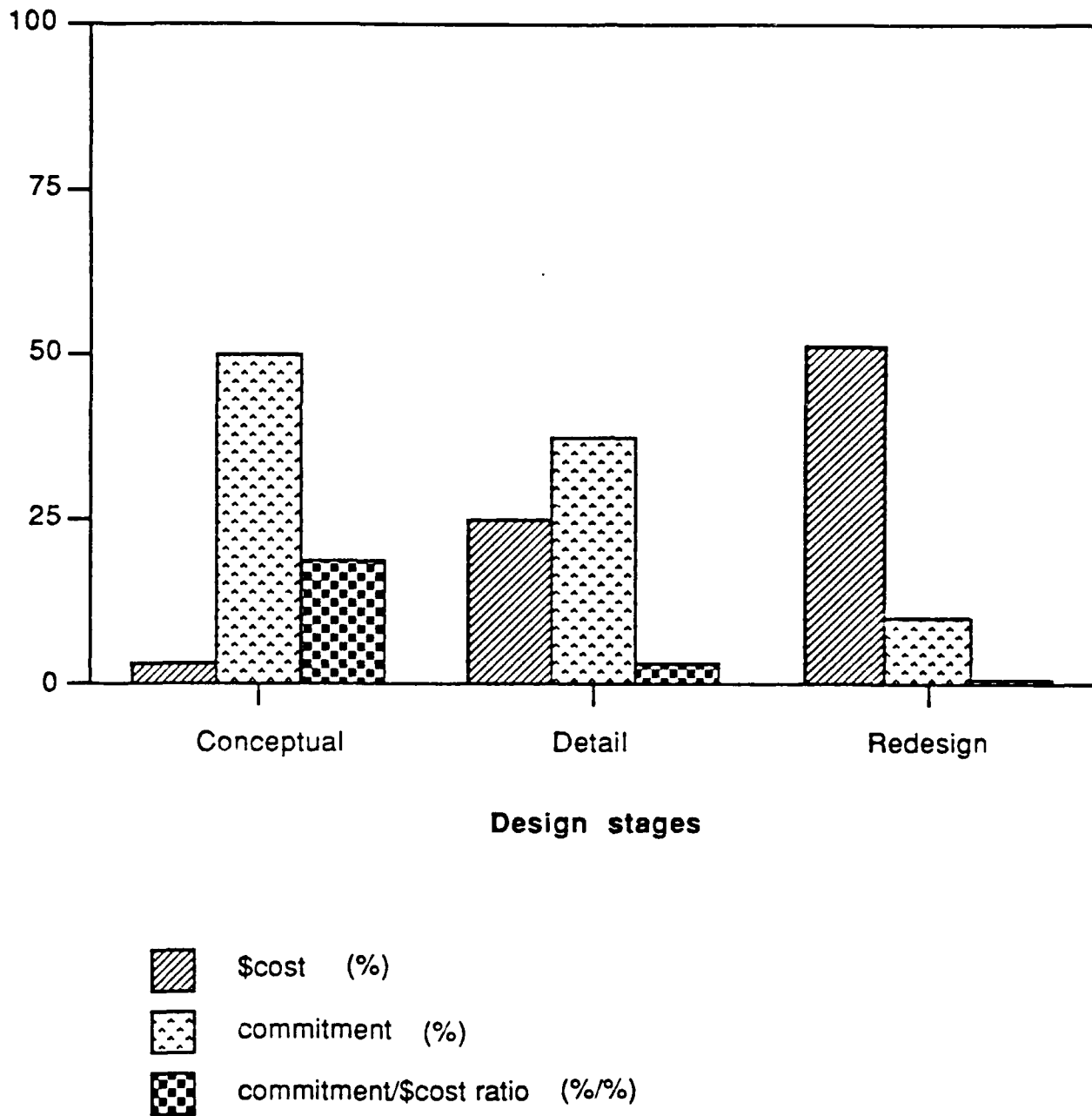


Figure 1.1 Current CAD/CAM environment.

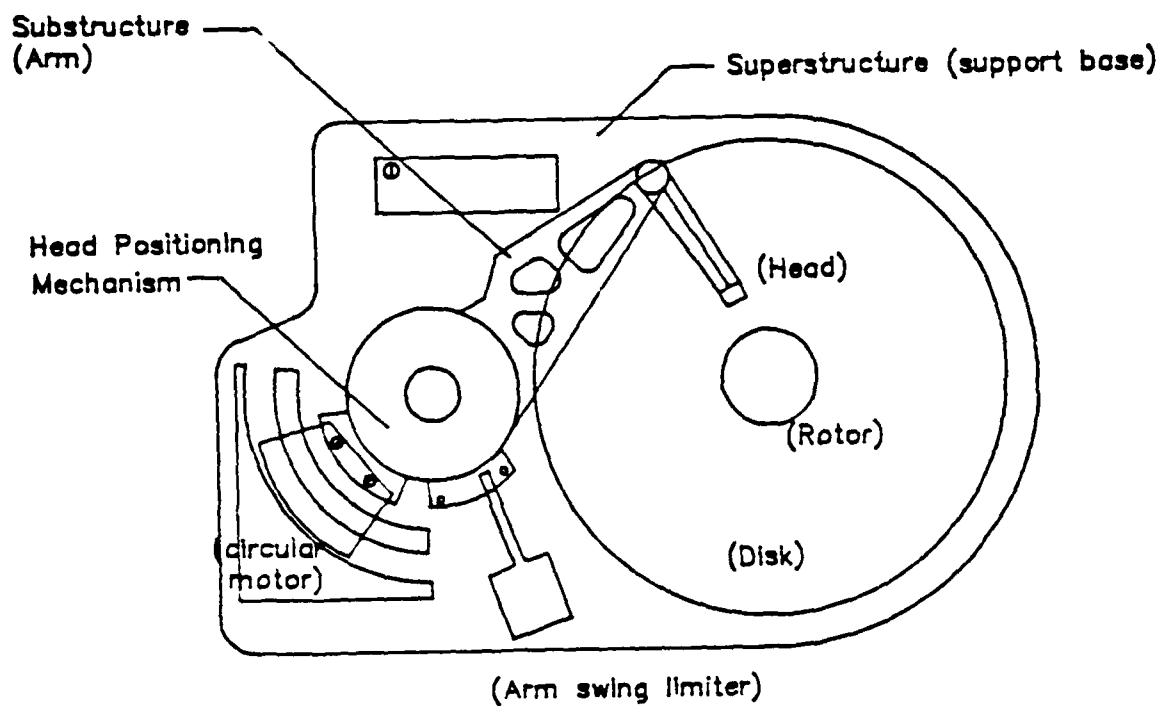


Figure 1.2 A Physical Representation of a Hard disk Drive.

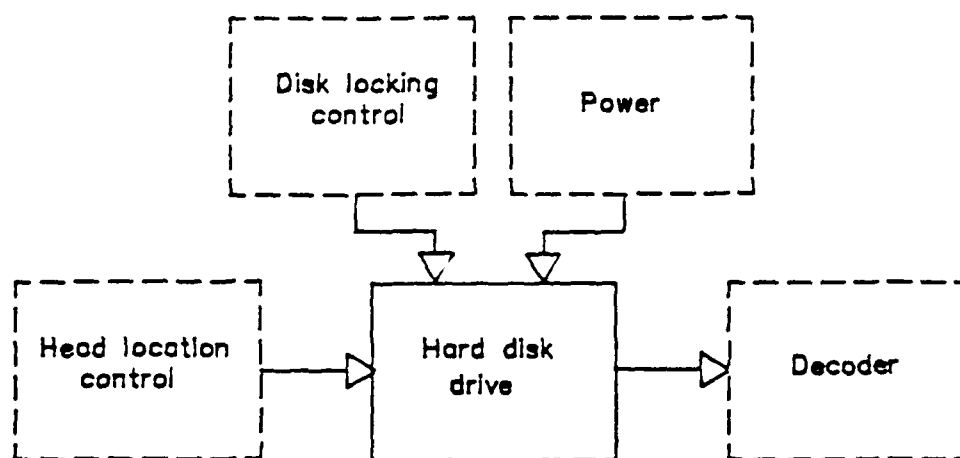


Figure 1.2 B. Conceptual Design - Functional Representation.

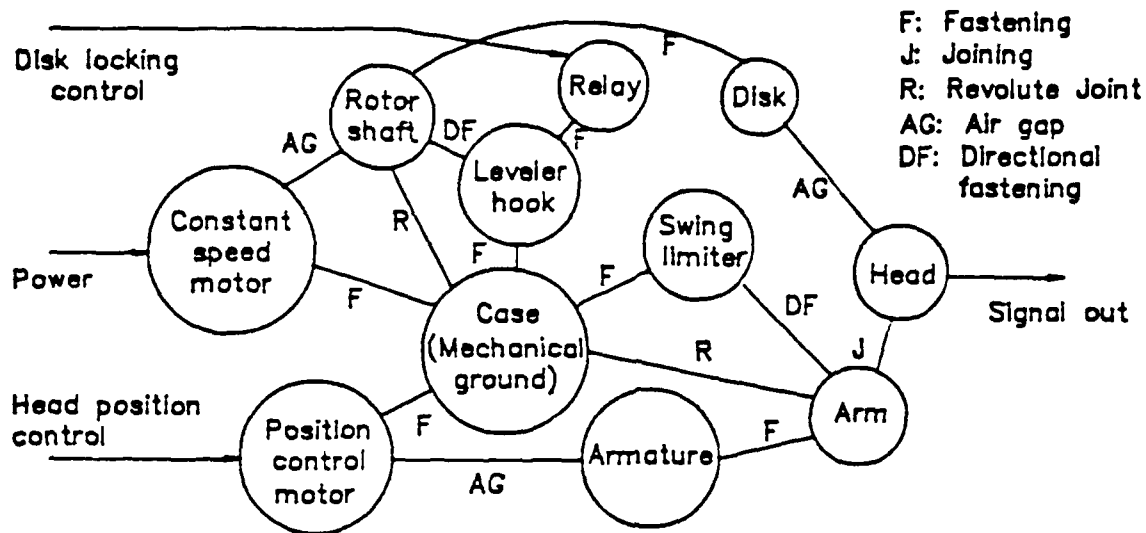


Figure 1.2 C. Conceptual Design - Connectivity Representation.

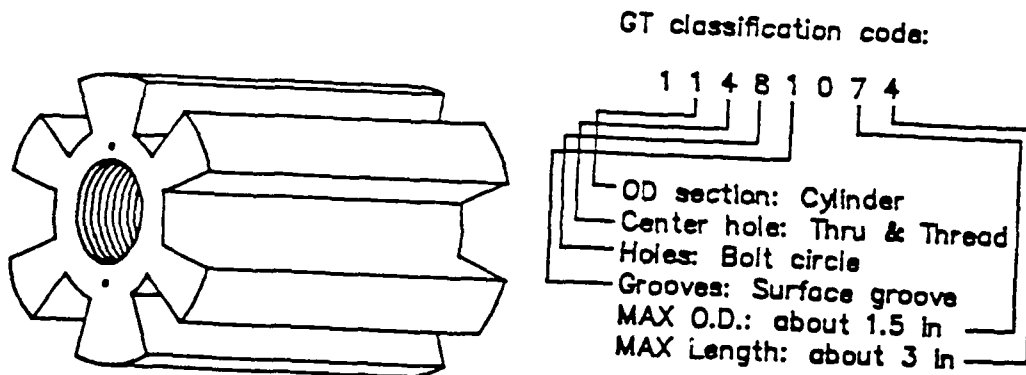


Figure 1.2 D. Conceptual Design - Component Concept (Group Technology),

2. The Relationship Between Conceptual Mechanism Design and Artificial Intelligence (AI)

- 2.1 The Conceptual Design of Mechanisms.
- 2.2 Intractability, Heuristics and Artificial Intelligence.
- 2.3 A Prototype Mechanism Design Expert System: MECXPERT.

2.1 The Conceptual Design of Mechanisms

Based on the application of a divide-and-conquer type of problem solution strategy, previous mechanism design researchers have consciously elected to separate mechanism design tasks into structural (conceptual design) and functional (detailed design) design processes [Freudenstein and Maki, '86]. During mechanism structural synthesis, an abstraction of a mechanism, which contains only the connectivity information of the mechanism, is included. This set of connectivity information is referred to as the kinematic structure of the mechanism. The kinematic structure can be stored in a compact mathematical form, as either a graph or a matrix (Figure 2.1). Within a kinematic structural representation, each node denotes a rigid body (link), and each edge denotes the connection type (joint type) which connects the links. While somewhat counter intuitive in nature (i.e., why not assign the links to the edges and the joints to the vertices of the graph), this method of assigning the structural entities of a mechanism to a graph has important implications.

Based on this method of representing kinematic structure, the conceptual design of mechanisms is transformed into a graph enumeration problem [Dobrzanskyj, '66], whereby different graphs are used to represent different types of mechanisms for various design purposes. However, a solution is not immediately at hand, since the mechanism design methodology, in requiring the creation of a graph, is considered to be a "hard" problem, and is recognized by Garey [Garey, '79] to possess inherent intractability. As a result of this, AI techniques have been brought to bear on these problems in order to enumerate potential solutions.

Function
 Function generation
 Rotary input \longrightarrow Reciprocating output

Structure
 Four links, four joints, input joint is R type,
 output joint is P type, one degree of freedom
 mechanism design

Mechanism
 Reciprocating
 output

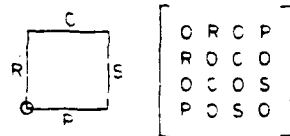
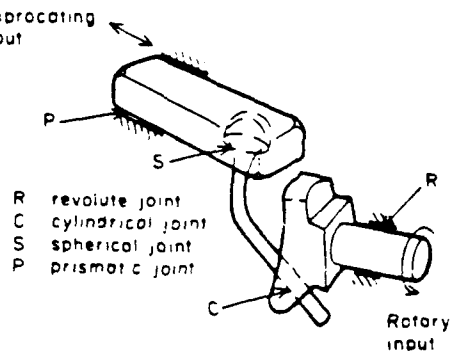


Figure 2.1 Separation of Kinematic structure from Function,
 Physical Representation, Graphs and Matrix Representations.

2.2 Intractability, Heuristics and Artificial Intelligence

AI techniques are necessary if one expects to obtain solutions to problems which demonstrate inherent numerical intractability. Such problems are referred to as NP-complete (NPC), or Nondeterministic Polynomial Time Complete [Gary, '79] problems. The cpu time required to solve an NP-complete problem, based on known algorithms, grows exponentially with the "size" of the problem. There exist no polynomial time transformations for NPC problems nor are there any polynomial time algorithms capable of solving any NP problems, therefore these problems are considered to be "open" or unsolved problems.

The potential to solve these NP and NPC problems depends on the availability of certain *heuristics*. As an example, minimization of the completion time for m parallel machines processing n jobs, $P||C_{\max}$, is known to be an intractable, NP-Complete problem. The heuristic which states that "jobs are added to the schedule in order of nondecreasing job processing time, J_j ", is known as the LPT heuristic [Lawler, '82], and has been analyzed as follows.

$$C(LPT)_{\max} / C(OPT)_{\max} \leq 4/3 - 1/3m$$

where $C(\star)$ denotes the required completion time based on the chosen algorithm, \star .

This states that in the worst case, the LPT heuristic can produce a decrease in the completion time by only one-third, as compared with the theoretical optimum solution (OPT). However, the OPT solution has either not been discovered, or it may be impossible to obtain.

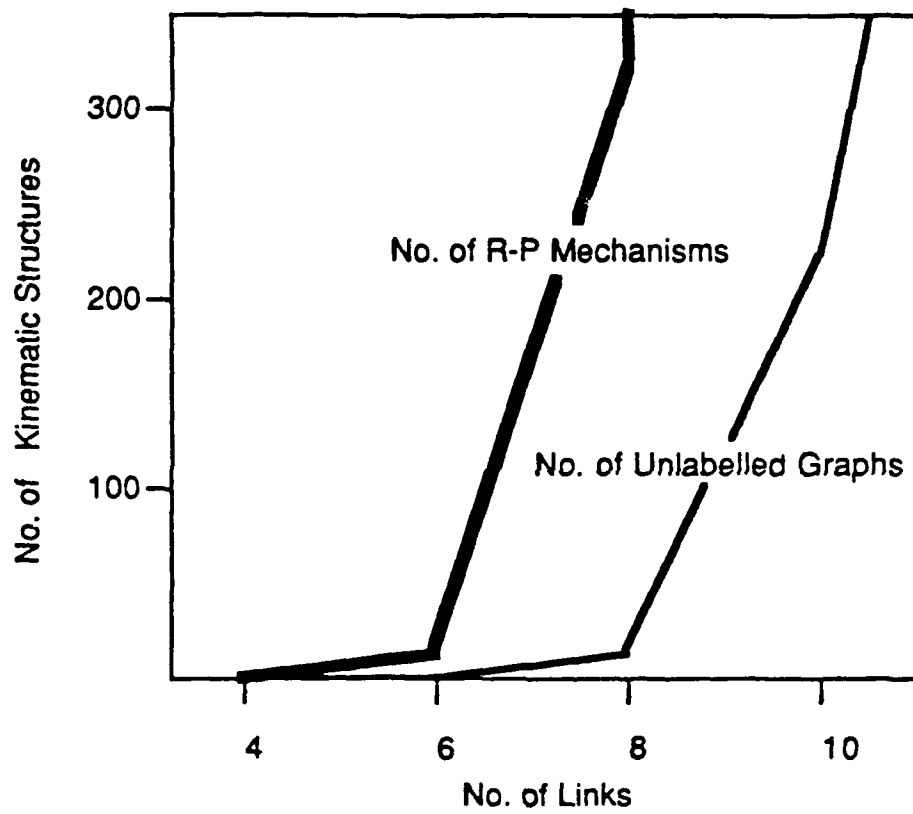
Applying optimization theory to the creative design of mechanisms poses an NP-Complete problem since a) graph enumeration is known to be an NPC problem (the number of graphs increase exponentially with the number of mechanism links required (this is shown in Figure 2.2), and b) graph isomorphism is an NP-Complete problem. Creative mechanism design is inherently intractable from the standpoint of NP-Completeness, therefore it requires design heuristics if solvability is to be expected.

Traditional, procedurally-based programming environments, based on the Basic, Pascal and Fortran languages, for example, offer very poor capability

for encoding design "heuristic knowledge". In order to transcend this computer programming language bottleneck, symbolic programming techniques associated with AI are, of necessity, introduced in order to compensate for the shortcomings inherent in traditional languages, and to expand the prospective domain over which a solution to the creative mechanism design problem may be obtained.

2.3 A Prototype Mechanism Design Expert System: MECXPART

The objective of this proposal is to explore the use of knowledge-based and symbolic programming techniques for automating the processes found in conceptual mechanism design. A prototype mechanism design expert system, referred to as MECXPART, has been designed and implemented on a Symbolics 3640 AI Workstation. The MECXPART expert system embodies four major processors, (Figure 2.3). These processors include, (1) synthesis, (2) analysis, (3) redesign and optimization, and (4) a man-machine interface. By merging the knowledge base, which provides design heuristics and symbolic manipulations, which handle the combinatorial optimization process, with a traditional mechanism design methodology (numerically-based design of mechanisms), the process of computer-aided mechanism design becomes more powerful and sophisticated from the standpoint of both *computational efficiency* and *computational sufficiency*.



- Design Through Global Search is Intractable.
- Design "Heuristics" are Required.
- AI Techniques ●

Figure 2.2 Creative mechanism design is an NP-Complete problem.

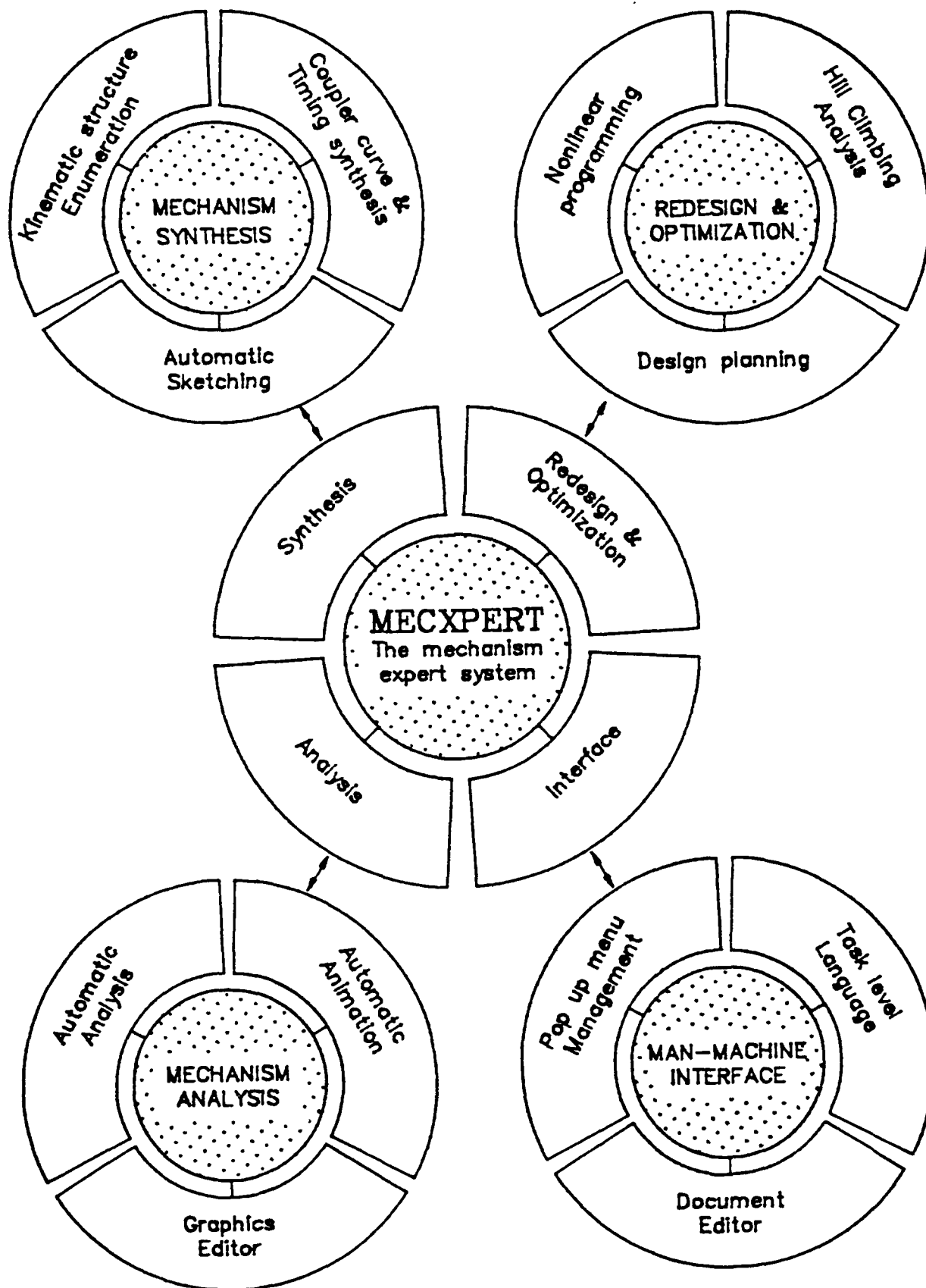


Figure 2.3

3. Tree Search Algorithms For the Enumeration of Design Concepts

3.1 Introduction to Tree Search.

3.2 The Enumeration of Graphs Corresponding to Kinematic Structure.

3.3 Automatic Sketching.

3.1 Introduction to Tree Search

It has been shown, based on a number of design cases studies, that through properly posing a design problem, the process of *integrated design decision making* can be represented as a decision tree [Aho, '83], the branches of which contain *subdesign decisions*. For example, a power transmission design can be broken down into subdesigns that include the design of speed reducers, clutches, motion converters, etc. Furthermore, the design of a speed reducer can be broken down into subdesigns corresponding to different types of speed reducers, such as gears, chains, etc. The design solution embodied in this tree structure takes the form of a path along the branches (the design solution is embedded in the solution process, eg. corresponding to design planning for a power transmission system, selected transmission components will constrain the selection of other transmission components; the solution of transmission design layout is rational, and therefore the design solution must not only be represented as individual components but also through component relationships and dependencies), or may simply be an individual leaf of the tree (design is actually the optimal selection of a simple solution, eg. selection of a known input/output, specific type of gear pairs). In either design problem, a tree search process is unavoidable, in order to obtain a design solution.

The three most commonly employed tree search strategies are depth-first, breath-first and best-first searches [Charniak, '85]. These are shown in see Figure 3.1. Depth-first search is usually applied to design optimization problems in which the search tree is broad but shallow. Breath-first search is usually employed in design optimization problems where the search tree is narrow. Best-first (or heuristic) search, is typically employed in design problems which are not only broad but also deep. During the search, state evaluation is performed by means of a heuristic measurement scheme, and is therefore not guaranteed to be accurate. While it does not guarantee an accurate solution, it is oftentimes better than simply relying on chance to yield a viable solution.

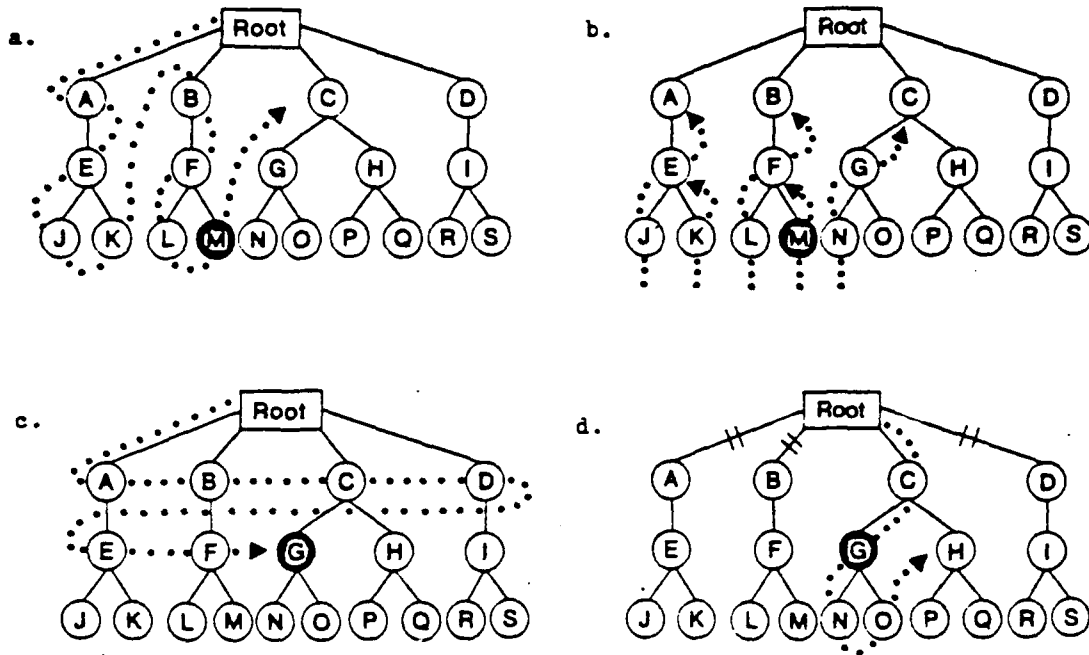


Figure 3.1 a. Forward Chaining / Depth-first search
 b. Backward Chaining
 c. Forward Chaining / Breath-first search
 d. Forward Chaining / Heuristic Search

[This Figure has been adopted from the cover of *Software for Engineering Workstations*, Vol.4 No.3, 1988, Computational Mechanics Pub. Co.]

3.2 Kinematic Structural Graph Enumeration

Graph representations have been employed to represent the kinematic structure of mechanisms, both for mathematical and computational efficiency purposes. The process of creating, listing and displaying varieties of mechanisms based on graph theory is referred to as graph enumeration. The process of graph enumeration can be further subdivided into monochrome and colored enumeration. While a monochrome graph (unlabelled graph) is capable of representing only the connectivity of a mechanism, colors corresponding to the individual edges of a graph (i.e., a labelled graph) can specify the types of joints corresponding to the kinematic structure. This is shown in Figure 3.2. A brute force method for generating the unlabelled kinematic structures of mechanisms is outlined below.

Function: Brute-force-unlabelled-graph-enumeration-algorithm

Input: l , the total number of links.

j , the total number of joints.

Output: The *nonisomorphic*, unlabelled, link adjacency list U , which implicitly contains the kinematic structure for a mechanism with l links and j joints.

Procedures:

Step 1. Initialize a list U with length $l(l-1)/2$ with 0's.

$$U = (0 \ 0 \ 0 \ 0 \ \dots)_{l(l-1)/2 \times 1}$$

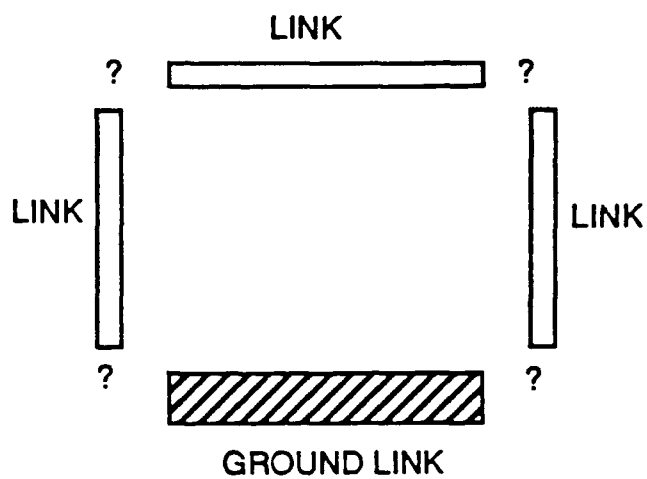
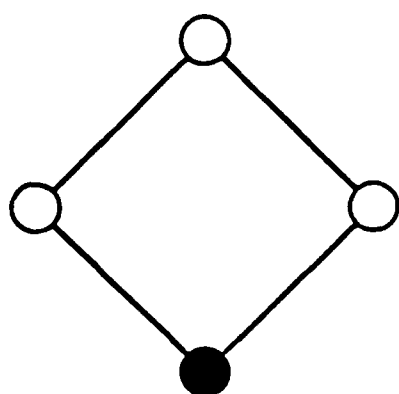
Initialize an $n \times n$ identity matrix, say $[M]$, with 1's stored in the diagonal elements.

Step 2. Apply DFS (depth-first search) to the 1's inserted in the list U , such that the total number of 1's in U is exactly j .

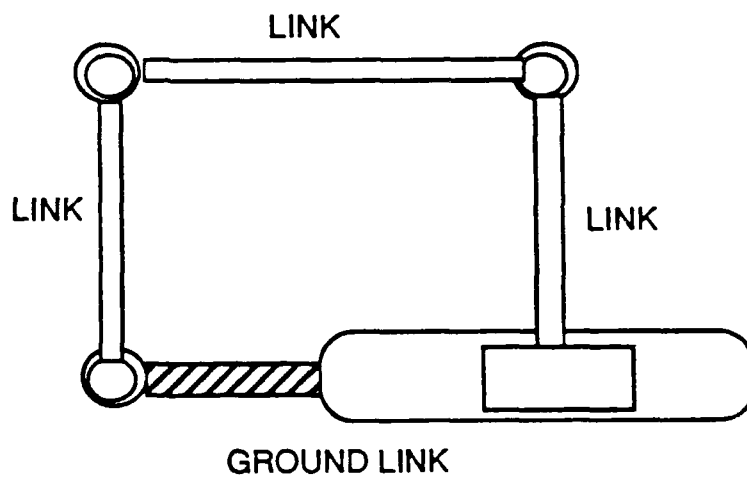
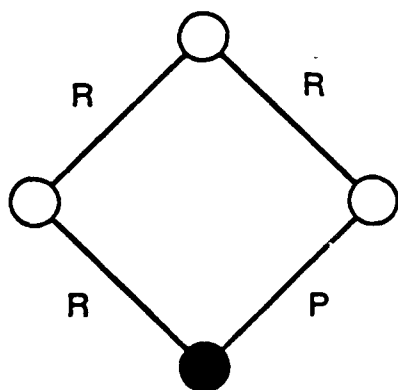
Since the link adjacency matrix M must be symmetric, one can load the one dimensional array U into the square matrix, M , by $M(r,s) = M(s,r) = U(rxj - j + s)$. where j is the total number of joints.

Step 3. If $\sum_{s=1} M(r,s) < 2$ for $r = 1, l$ then kinematic structures this category contains underconstrained links which are undesirable, go to Step 2.

Elseif Total number of paths from $M(r,s)$ to $M(q,t) \leq 1$



(A) Implications of an unlabelled graph.



(B) A labelled graph and its corresponding kinematic structure.

Figure 3.2

for $0 < r, s, q, t \leq l$ and $r \neq q$ and $s \neq t$

then the kinematic structures in this category will not be totally controllable by any chosen input. These are referred to as cut vertices in the graph, which are undesirable, go to Step 2.

Else go to Step 4.

Step 4. Retrieve [A]'s, the existing unlabelled kinematic structures from the unlabelled graph database and compare them with [M], using the characteristic polynomial method [Yan '83].

If $\text{determinant}([A] - X[I]) = \text{determinant}([M] - X[I])$

then goto step 2.

Step 5. Store the associated link adjacency list U, in data base as a new unlabeled kinematic structure with l links and j joints found.

Problems exist with brute force enumeration methods. One such problem lies within step 3 of the Brute-force-unlabelled-graph-enumeration algorithm. This step contains a redundancy which can potentially generate a large number of isomorphic graphs, and is therefore inefficient. The other problem lies within step 4 of the Brute-force-unlabelled-graph-enumeration algorithm. In this step, application of the characteristic polynomial method for verification of graph isomorphism has been proven to be insufficient. Modifications which have been made to the solutions of these problems have been made during the past five years, and are discussed below.

1. Optimal-code isomorphism identification [Shah, '74]: Instead of applying the characteristic polynomial method, the optimum code method suggests permuting the vertex labelling of the graph, so that the binary sequence of the upper triangular adjacency matrix becomes a maximum or minimum. This allows verification to be performed through checking of the optimum code.
2. Dual representation of graphs and mechanisms [Sohn, '87]: Step 3 in the brute-force-unlabelled-graph-enumeration algorithm is rather computationally expensive (i.e., it is inefficient). Sohn and Freudenstein have applied dual graph concepts to enumerate planar graphs as well as mechanisms in a non-duplicating sense, and have improved the efficiency.
3. Stratified Representation Approach [Fang, '89]: Fang and Freudenstein

have designed a hierarchical scheme for enumerating graphs. This new method improves both the efficiency and sufficiency for both unlabelled and labelled graph enumeration.

3.3 Automatic Sketching

Sketching the graphs of mechanisms through the assignment of "appropriate" dimensions which satisfy the connectivity specification (graph), appears to be a computationally intensive problem. The specification of "appropriate" dimensions for sketching usually requires algorithmic control based on the satisfaction of constraints which 1) prevent the existence of crossing links [Woo, '69] (the links are not permitted to overlap or crossover one another other for the sketch of a planar mechanism) and 2) require some minimally sufficient length length aspect ratio (the ratio of the lengths of the maximum-to-minimum link lengths).

A number of investigators, including [Olson, '84] and [Chieng and Hoeltzel, '88a] have developed algorithms which represent significant attempts at solving the automatic sketching problem. Automatic sketching of planar linkages can be mathematically formulated as follows.

Function: Automatic-Sketching-of-Planar-Mechanisms

Input: Kinematic structure, a labelled link connectivity matrix with j joints and l links.

Variables: The location and orientation of each joint, denoted as j_x, j_y for an R (revolute) joint and j_x, j_θ for a P (prismatic) joint.

Define: L_k as the k th link length.

Formulation of the optimization problem:

OBJ: Minimize $(\text{Max}\{L_k\} / \text{Min}\{L_k\})$ {Aspect ratio}
 $k=1, l \quad k=1, l$

Subject to the constraints: No two links are permitted to cross.

The link crossing condition can be identified through a geometric intersection test. The result of automatic sketching of planar mechanisms can be seen in Figure 3.3.

Besides the aspect ratio optimization criterion, an additional practical design consideration in mechanism sketching concerns the satisfaction of a rotatability criterion. Under this condition, the rotary input must be

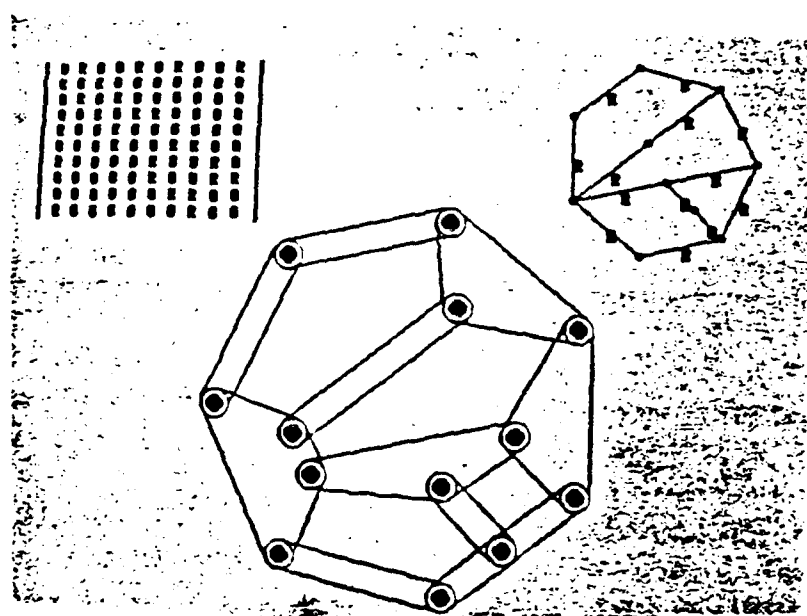
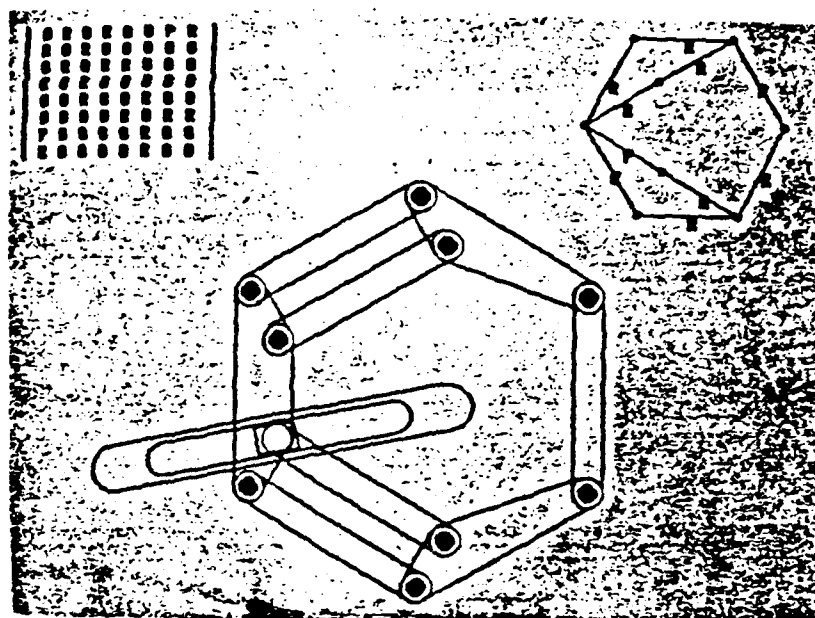


Figure 3.3 Automatic sketching of planar kinematic chains.
Top : Sketch of an eight-link kinematic chain.
Bottom: Sketch of a ten-link kinematic chain.

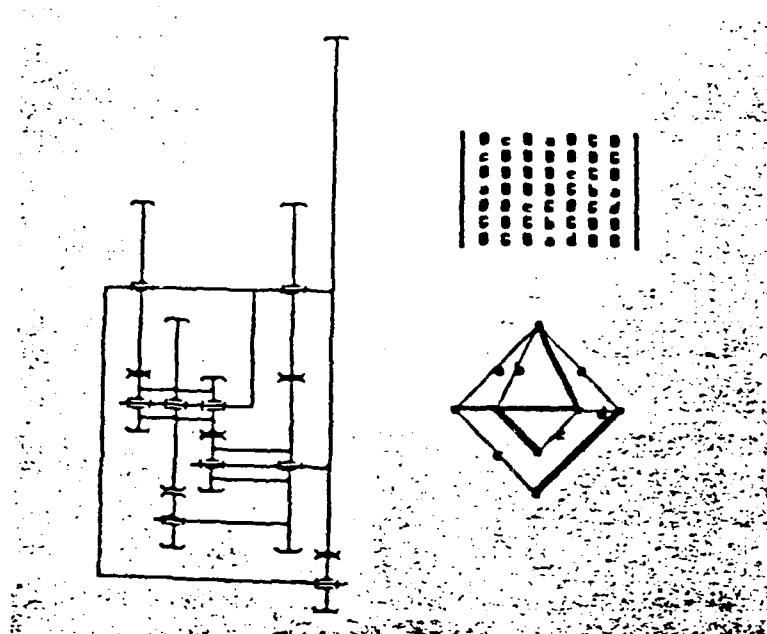
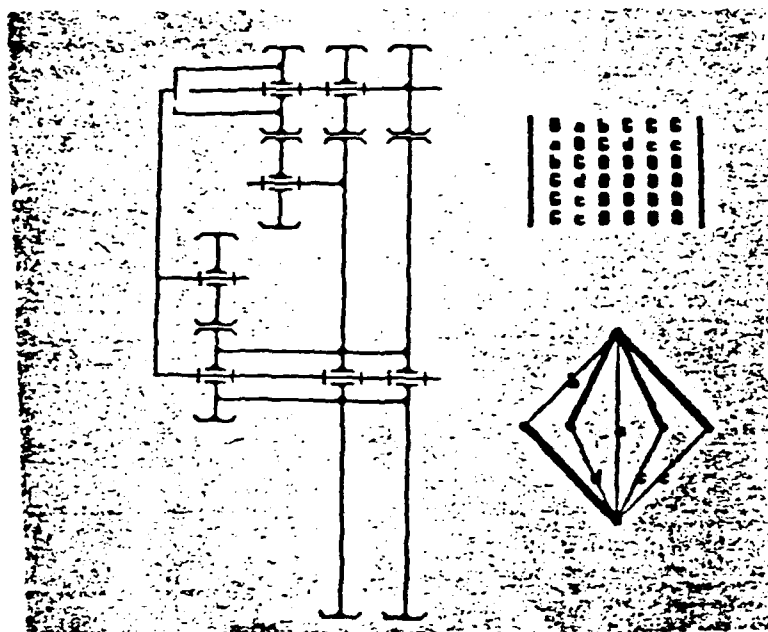


Figure 3.3 (continued)

Automatic sketching of epicyclic gear trains.

Top: Sketch of a gear train with 4 gear pairs and 6 links.

Bottom : Sketch of a gear train with 5 gear pairs and 8 links.

allowed to rotate through a complete 360° cycle. Based on practical observations, the number of arithmetic operations, i.e. +, - * /, in the kinematic equations for verification of rotatability grows exponentially with the number of links. It does not seem feasible, based on currently available computing power, to be able to symbolically derive the kinematic constraints which satisfy the rotatability conditions required in mechanism sketching without human intervention.

4. Knowledge Representation and Planning Control For Mechanism Design

- 4.1 Introduction to Knowledge Representation.
- 4.2 Introduction to Rule-Based Expert Systems.
- 4.3 Planning Control for Conceptual Mechanism Design.

4.1 Introduction to Knowledge Representation

A frame-based knowledge representation scheme employs a slot-filler type of knowledge structure to store knowledge about an object [Tanimoto, '87]. Maintenance of inheritance and **transparency** between objects, throughout a hierarchical structure, is the primary task to be satisfied in this knowledge representation scheme.

A semantic network-based knowledge representation scheme employs a graph-based representation (tree-structure) to express relations among different objects [Tanimoto, '87]. Creation and retrieval of the network relations is the primary task to be satisfied by this type of knowledge representation scheme. Figure 4.1 demonstrates these two ways of representing knowledge within the domain of mechanism design.

Procedural and control knowledge are commonly expressed in the form of antecedent-consequence pairs of IF-THEN rules. They may be deterministic, in nature, such as in the case of,

If (*antecedent-1, antecedent-2, antecedent3,*)
 then (*consequence-1, consequence-2,*)

or, they may express nondeterministic knowledge using fuzzy logic, i.e.

If expectation-of (*antecedent-1, antecedent-2, antecedent3,*)
 is greater than the certainty factor threshold <cf1>
 then conclude (*consequence-1, consequence-2,*)
 with certainty factor <cf2>
 where <cf> denotes the certainty factor.

4.2 Introduction to Rule-Based Expert Systems

A typical design expert system contains two primary parts [Brownston, et. al., '85]:

1. Memory: Based on the knowledge representation methods described

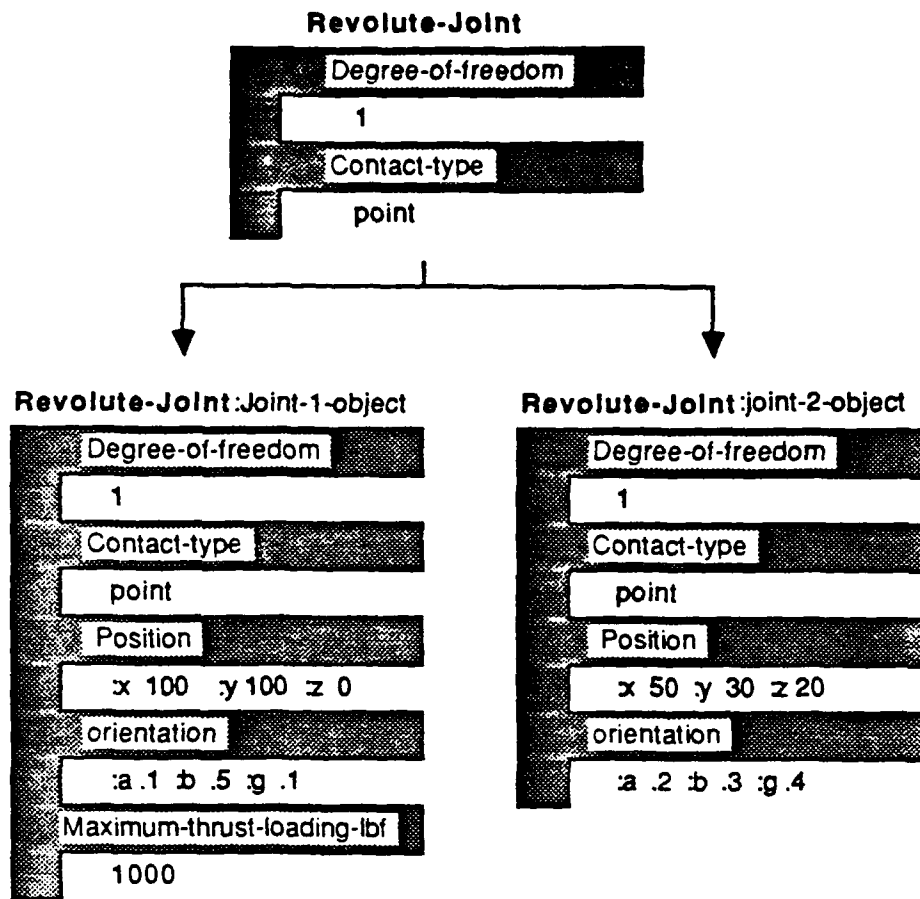


Figure 4.1 A. Frame-based knowledge representation scheme.

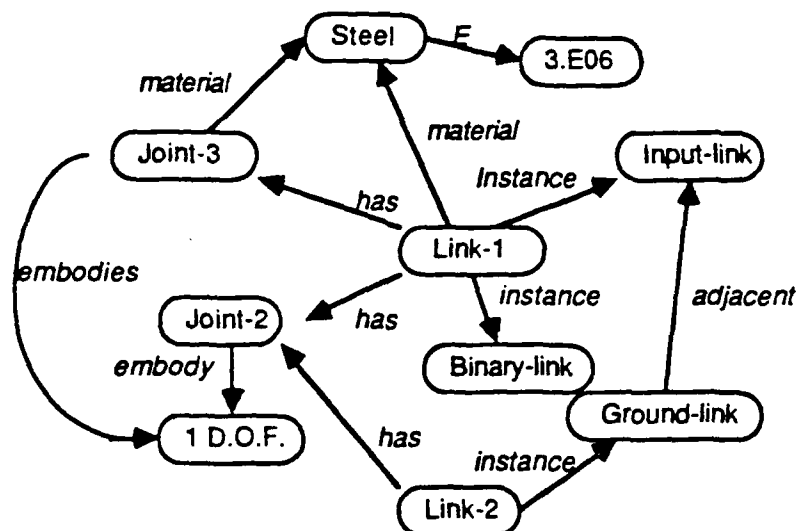


Figure 4.1 B. Semantic network-based knowledge representation scheme.

4.3 Planning Control in Mechanism Design

In a "plain", rule-based expert system (*production system*) , it is necessary to perform rule matching on all the rules stored in *production memory* against all the data which resides in *working memory* (Figure 4.2). The overall efficiency of such an expert system can become severely hampered when it is required to operate on a large knowledge base, which is typically the case for real engineering design problems.

Given the current level of applied AI technology, it is not feasible to directly apply a "plain", or pure heuristically-based expert system to attempt to successfully automate the process of creative mechanism design. In order to increase system efficiency, a *hybrid* (symbolic-numeric) *design system* is required (described in greater detail in Chapter 8), whereby partitioning of both the long-term working memory, which contains permanent facts, and production memory, which contains design rules, is performed.

Design planning control [Brown, '86] represents the means through which communication is established among the partitioned modules in a hybrid design system. Each partitioned module containing a set of rules and local long-term memory is called a design plan. The critical problem to be addressed regarding intermodule communication, is the technique employed to optimally sequence the design plans.

Important factors to be regulated in the controlled sequencing of design plans have been analyzed by [Hoeltzel and Chieng, '87, '88a], and include (Figure 4.3):

1. State evaluation: Evaluate the condition of the current design state using predefined *operators*.
2. Sequencing and scheduling: Based on indicators acquired from *state variables*, determine the next design process.
3. Constraint propagation: The manner in which information is permitted to flow between the different partitioned memory modules (design plans).

The Mechanism Expert System {MECXPRT} contains a design planning

Deterministic Rule:

If (*antecedent-1, antecedent-2, antecedent-3,*)
Then (*consequence-1, consequence-2, consequence-3, ...*)

eg.

If (Instance <Loop>
 ^Is-a Independent-Loop
 ^belongs-to <mechanism>
 ^joint-list <jl>) and
 (evaluate (> (number-of-slider-joints <jl>) 2))
then (Instance <mechanism>
 ^validity failure
 ^reason harmful-mobility)

{If there are more than two slider joints in any single loop then
 the topology is invalid}.

Fuzzy Logic Rule:

If expected-value-of (*antecedent-1, antecedent-2, antecedent-3,*)
 is greater than a certainty factor threshold <cf1>
Then
 conclude (*consequence-1, consequence-2, consequence-3, ...*)
 with certainty factor <cf2>

Figure 4.1C IF-THEN rule-based knowledge representation scheme.

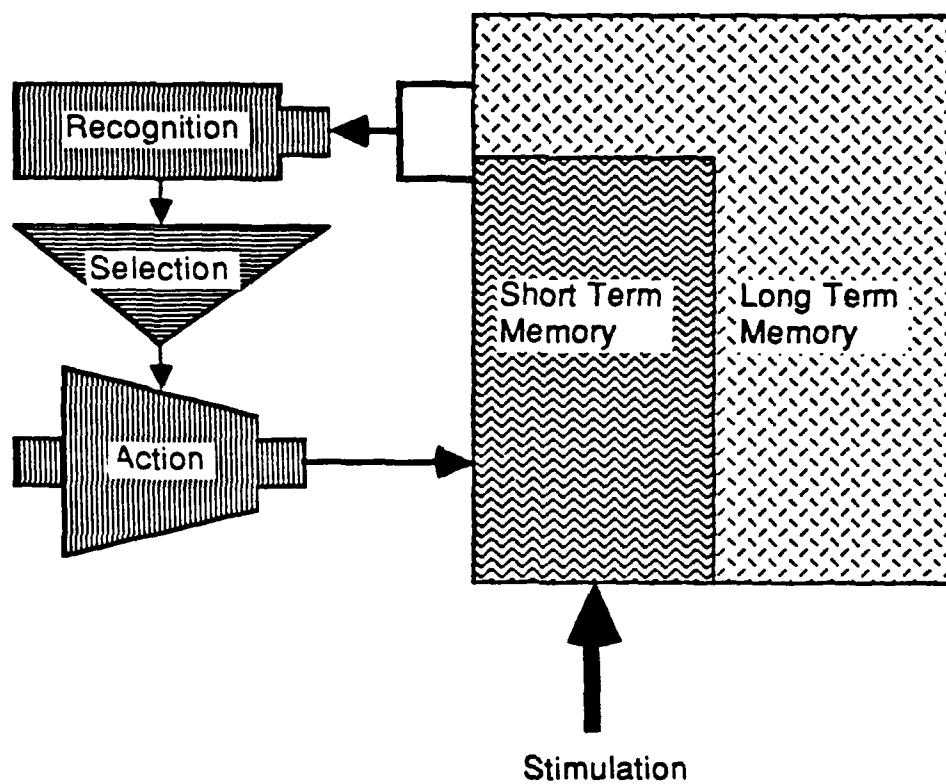


Figure 4.2 Inferencing process within an expert system

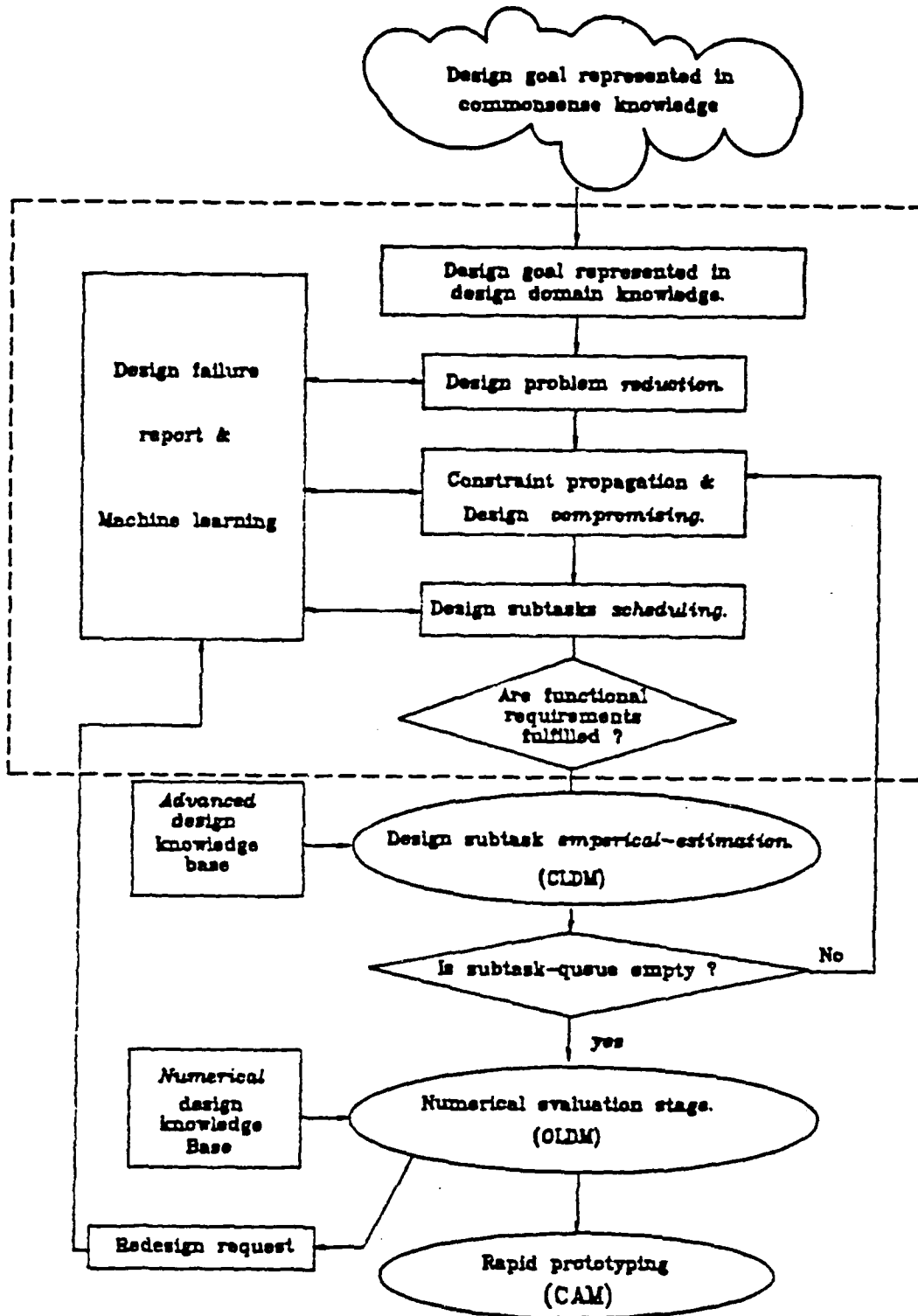


Figure 4.3 A generic design planning scheme.

process for creative mechanism design which adopts a state space representation scheme for controlling the design process. A design planning control system for the creative design of mechanisms is shown in Figure 4.4. The results of applying this planning control scheme to the design of a variable-stroke engine mechanism is shown in Figure 4.5.

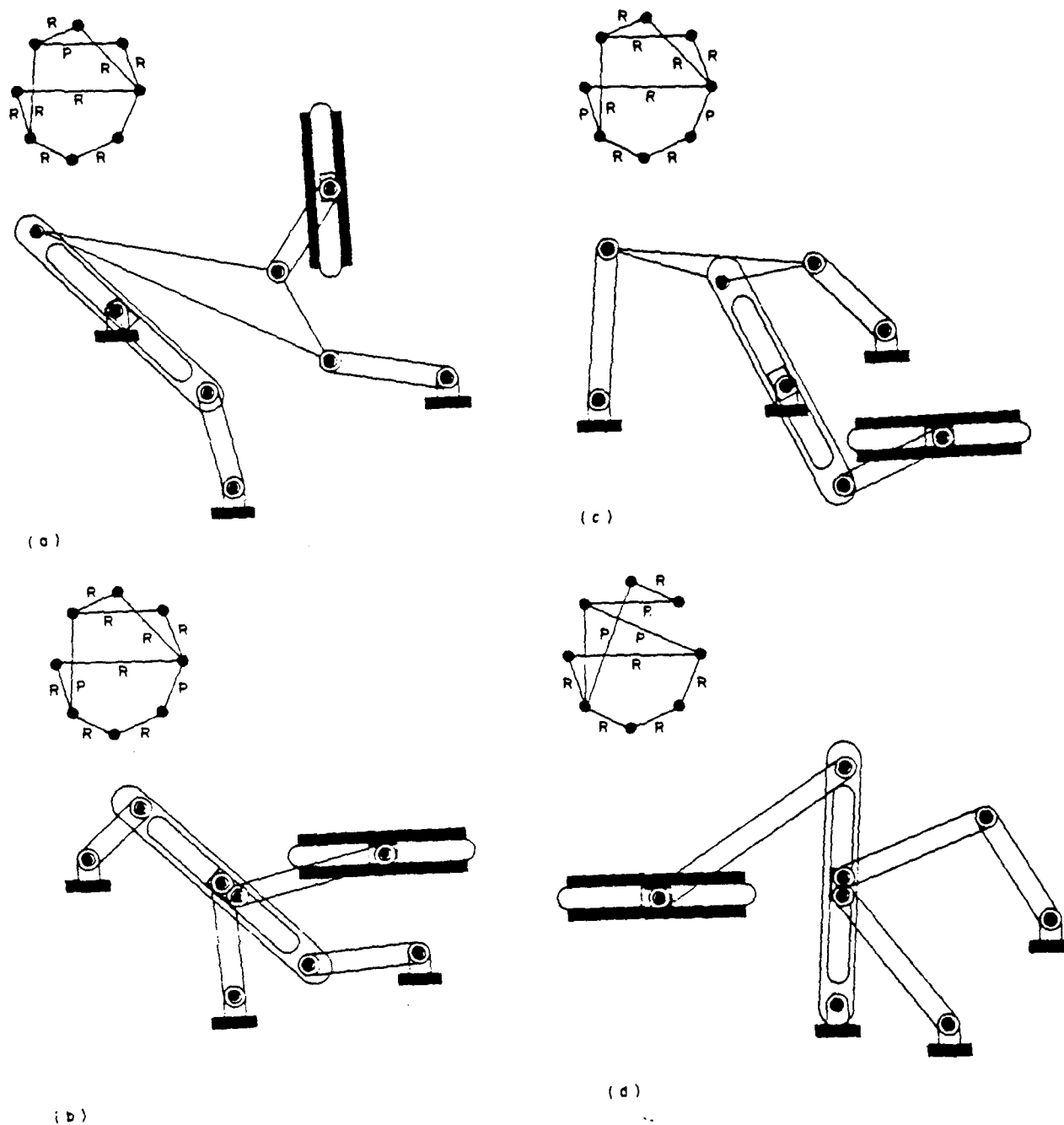


Figure 4.5 Partial design output generated by the design planning process. Variable stroke engine mechanism design: (a), (b) and (c) are generated from the same unlabelled graph. (c) and (d) are generated from different unlabelled graphs.

5 Clustering Analysis in Mechanism Design

5.1 Introduction to Clustering Analysis.

5.2 Introduction to Neural Network Computing.

5.3 The Creative Synthesis of Mechanisms Based on Clustering Analysis
Applied to Coupler Curve Images (Pattern Matching Synthesis).

5.1 Introduction to Clustering Analysis

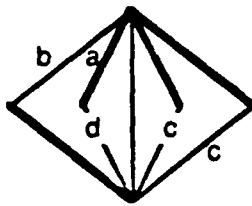
Clustering is defined to be the unsupervised classification of a set of data [Bow, '84]. It is the process of generating classes without any a priori (expected) knowledge of prototype classification. As an example, consider the conceptual design of a gear train, generated using the graph enumeration process, subject to the constraint that it must be inserted into a space of limited size. A Monte Carlo simulation can be applied to assign the positions of the gear shafts and the gear ratios for each gear pair, subject to the constraints that a) the maximum gear ratio for each pair is seven, b) the fatigue strength corresponding to a predetermined gear tooth life is specified, and c) the gears cannot be permitted to interfere with each other. A set of results for this problem is shown in Figure 5.1. The x-axis displays the length/height ratio of the gear box and the y-axis displays the Input/Output gear ratio.

Clustering analysis is able to draw conclusions on the results, in this case of a Monte Carlo simulation, about the design of a gear train, without the need for human intervention. This process is similar to that found in generalized learning [Michalski, '86], not unlike that performed by a human being.

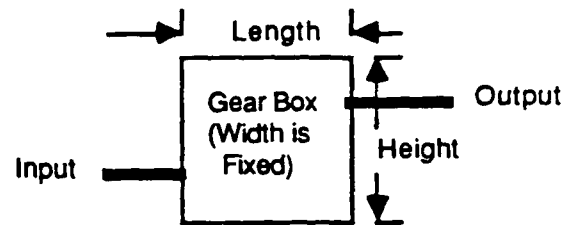
Based on the calculation of the Euclidean distance, denoted as D_{ij} ,

$$D_{ij} = \left[\sum_{k=1}^n |z_{ki} - z_{kj}|^2 \right]^{\frac{1}{2}}$$

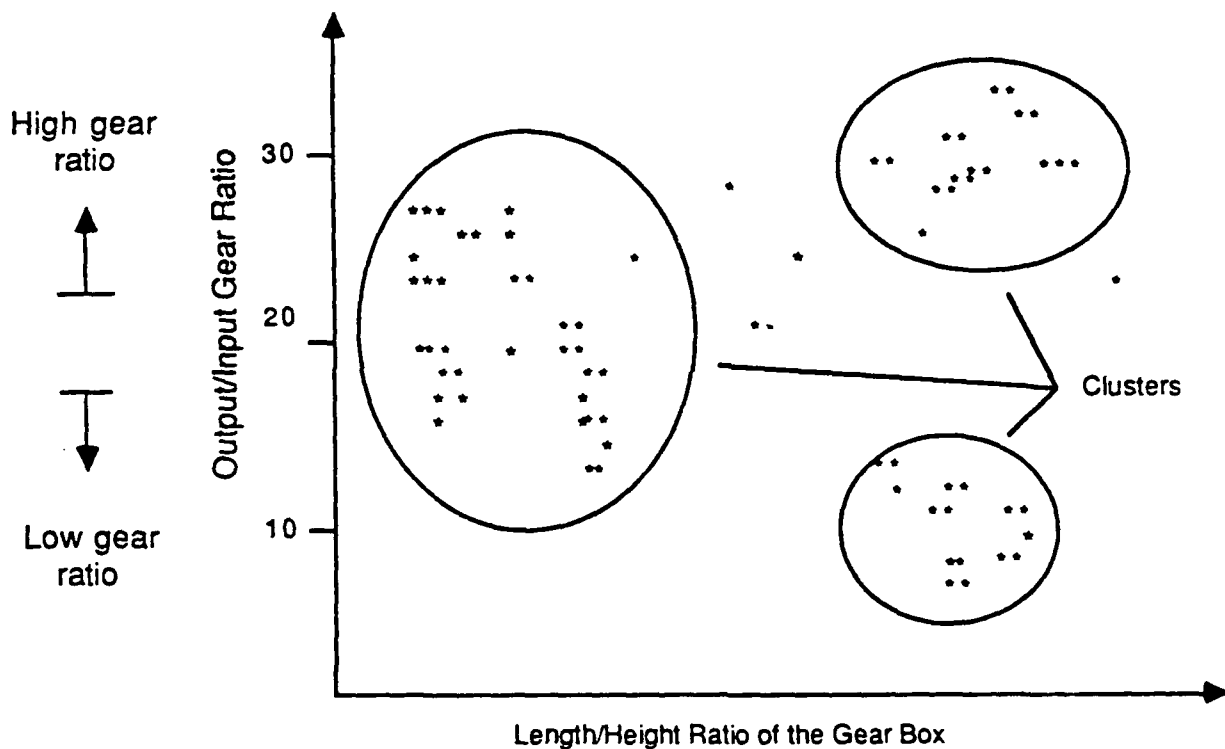
where z_{ki} is the k th attribute for object i . Non-hierarchical clustering analysis attempts to group the data instances into classes, while both minimizing the variance within each group and maximizing the average Euclidean distance between the centroids of the different groups. An efficient algorithm for applying clustering analysis, having an algorithmic complexity $O(m^3+n^3)$, where m is the minimum number of groups and n is the number



Graph representation for the
Conceptual Design of a Gear Train



Parametric Design of a Gear Box



Observations:

1. When the Length/Height ratio is low, the output/input gear ratio is medium.
2. Missing knowledge required to determine the design of a high or low output/input gear ratio when the length/height is high.

Figure 5.1 Clustering analysis applied to a Monte Carlo simulation for the conceptual design of a gear train.

of objects. has been developed by [Hoeltzel and Chieng, '88].

5.2 Introduction to Neural Network Computing

Unsupervised clustering analysis divides an entire space comprised of n data instances into m distinct cluster groups (classes). The pertinent question to be answered from clustering analysis is, what are the attributes associated with each cluster group? For example, if one wants to design a gear train having a low overall gear ratio, subject to a space constraint of low length/height ratio, as shown in Figure 5.1, a neural network (NN) learning process can be used to extract the basic rules to be followed by the designer in successfully implementing such a design.

A number of neural network models (NNM) have been used for practical applications in the fields of pattern recognition or machine learning. The neural network employed here, focuses on the subject of machine learning. In developing a mathematical formulation for our NNM, we define a set, C_i , of instances, s , as $\{s \mid s \in C_i\}$, each instance of which has p attributes, $s = \{a_1, a_2, a_3, \dots, a_p\}$.

Two possible types of machine learning can be achieved through the use of neural network computing:

1. Classification: Find the minimum number of hyperplanes, or hypersurfaces in p -space (since there are p attributes), which are sufficient in number to separate the m cluster groups (classes), $C_1, C_2, C_3, \dots, C_m$. A multiple-layered committee machine [Hinton, '87] provides a scheme similar to that employed in the generation of Voroni diagrams [Preparata, '85], as an effective method for classification.
2. Generalization: Generalize the data instances within each cluster group according to the attributes that they share in common. During the attribute generalization process, correlations among the attributes can also be learned. An example is shown in Figure 5.2. The attributes are specified as nodes (neurons), and correlations between attributes are represented as connecting edges. Each common attribute for a cluster is called a "concept". Using neural network computing, a concept can be learned by feeding an entire set of instances belonging to a cluster, through the NN. This is referred to as generalization learning. The

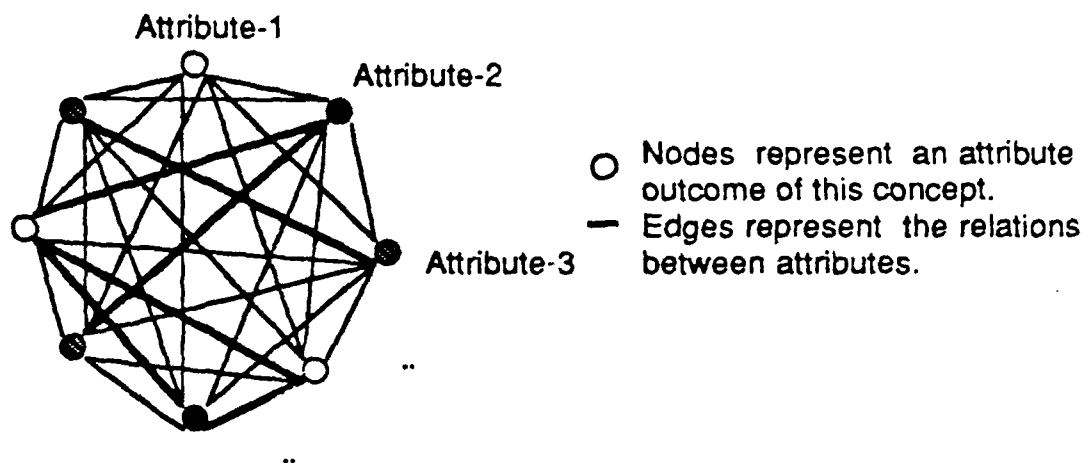


Figure 5.2 A. Neural Network for the representation of a "Concept".

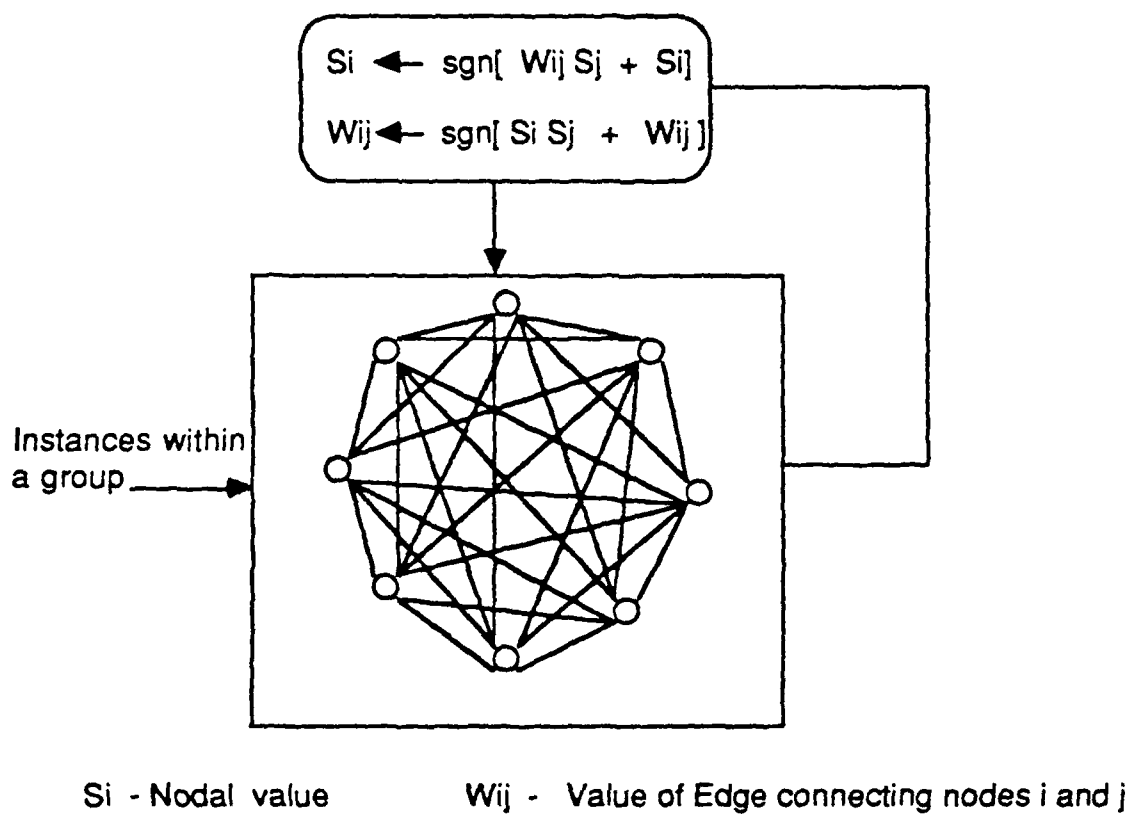


Figure 5.2 B. Training a concept using a Hopfield Neural Network Model.

Hopfield NN Model [Hopfield, '85], and other NN [Lippmann, '87] models have been successfully applied to generalization learning problems.

5.3 Clustering Analysis Applied to Mechanism Coupler Curves: {Pattern Matching Synthesis}

The methodology for creative mechanism design, based on graph enumeration, lacks sufficiency for the design of mechanisms, based on path synthesis, i.e., the mechanism to be designed must be able to pass through a predefined path, within a predefined tolerance (error) band. Current knowledge about analytical path synthesis is only sufficient to handle, in a complete manner, the four bar mechanism problem [Freudenstein, '59] [Kramer, '75].

The ambitious idea that "path design synthesis should be able to be carried out on a more general class of mechanisms" has been stated by Hoeltzel and Chieng [Hoeltzel, '88]. They have described a systematic method for generalizing mechanism path synthesis using clustering analysis. They refer to this method as Pattern Matching Synthesis (PMS). It is based on an approach which randomly generates coupler curves, using a parametric representation for the link lengths, as shown as Figure 5.3. By varying each of the link lengths of a mechanism, in a parametric manner, an entire series of coupler curves can be generated. In developing this approach, we have elected to use two ways of representing the attributes of a coupler curve:

1. Moment Invariant-based: Hu [Hu, '62] developed a moment calculation scheme. The moment invariants, up to 7th order, are denoted as $\phi = \{\phi_1, \phi_2, \dots, \phi_7\}$. These moment invariants allow for the elimination of differences between the images based on scaling, translation and rotational transformation differences of the curves. Hence, if two curves are the same except for scaling, translational and rotational transformations, they will have the same moment invariants ϕ .
2. Digital Image-based: Digitize the continuous coupler curve image into a 30 x 30 digital image. The image is represented as a 30 x 30, on/off binary number array, $[D]_{30 \times 30}$. The attributes of each coupler curve

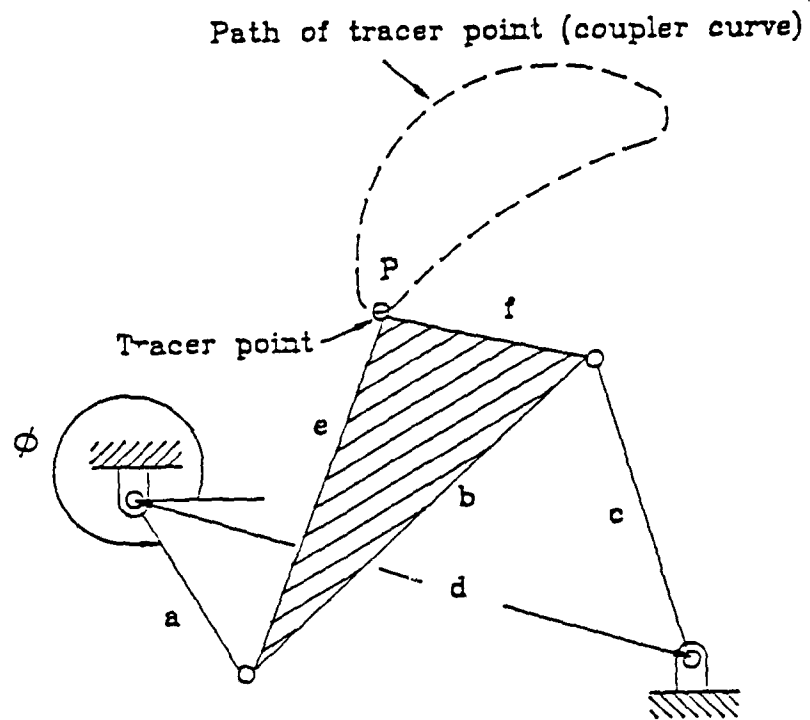


Figure 5.3 Systematically generate coupler curves for four-bar linkages, according to their link parameters (a , b , c , d , e and f).

- a - length of input link.
- b, e, f - lengths of edges of coupler link.
- c - length of output link.
- d - length of ground link.
- ϕ - input angle.

contains 900 elements, where $\{a\} = \{a_k \mid d_{ij} \text{ where } i = \text{int}(k/30) - 1, j = k - i*30, 1 \leq k \leq 900\}$.

The process of generating attributes is depicted in Figure 5.4. The procedures, which are based on prescriptions developed by Hoeltzel and Chieng [Hoeltzel, '88], for the entire clustering analysis, are described as follows.

- Step 1. Parameterize each link length from 0 to 1 (0.0, 0.33, 0.66, 1.0) and parameterize the position of the trace point on the coupler link.
- Step 2. Animate each mechanism generated from step (1) (i.e., having different link lengths and coupler points) and store the coupler curves generated from these mechanisms. Calculate the moment invariants and digitize the coupler curve.
- Step 3. Cluster the coupler curves into classes according to their moment invariance values.
- Step 4. Classify the coupler curve groups (classes) according to their digital images using a three-layered committee machine [Hoeltzel and Chieng, '88].
- Step 5. Learn the common attributes from their digital images using a Hopfield Neural Network Model.

Based on the process described above, we have generated 500 coupler curves from the parameterically specified mechanism configurations. Following clustering analysis, the resulting coupler curves have been classified into fifteen different groups (classes). The common attributes associated with each digital image are shown in Figure 5.5.

Backward matching of a user-specified coupler curve, for a given path matching problem, with coupler curves stored in the curve data base corresponds to a "pattern recognition" problem. This is embedded within a committee machine classification scheme. One simply needs to expose the desired coupler curve to all the curve groups (classes). If any of the curve groups response is "it is a curve within my group", then this group is said to be a "match" with the user-specified curve.

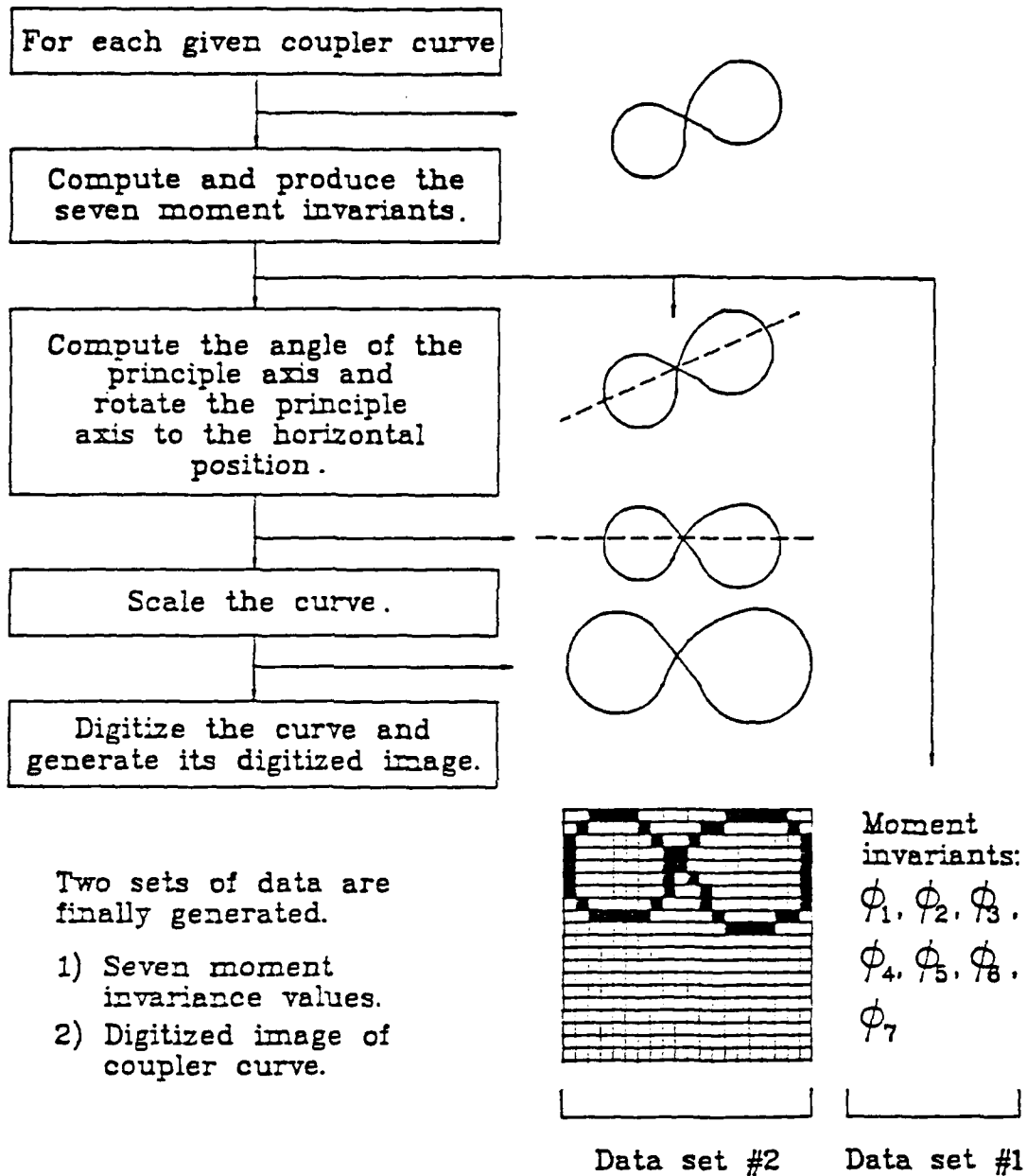


Figure 5.4 Scheme for acquiring data from the coupler curve of a given mechanism.

Chieng [Hoeltzel, '88], for the entire clustering analysis, are described as follows.

- Step 1. Parameterize each link length from 0 to 1 (0.0, 0.33, 0.66, 1.0) and parameterize the position of the trace point on the coupler link.
- Step 2. Animate each mechanism generated from step (1) (i.e., having different link lengths and coupler points) and store the coupler curves generated from these mechanisms. Calculate the moment invariants and digitize the coupler curve.
- Step 3. Cluster the coupler curves into classes according to their moment invariance values.
- Step 4. Classify the coupler curve groups (classes) according to their digital images using a three-layered committee machine [Hoeltzel and Chieng, '88].
- Step 5. Learn the common attributes from their digital images using a Hopfield Neural Network Model.

Based on the process described above, we have generated 500 coupler curves from the parameterically specified mechanism configurations. Following clustering analysis, the resulting coupler curves have been classified into fifteen different groups (classes). The common attributes associated with each digital image are shown in Figure 5.5.

Backward matching of a user-specified coupler curve, for a given path matching problem, with coupler curves stored in the curve data base corresponds to a "pattern recognition" problem. This is embedded within a committee machine classification scheme. One simply needs to expose the desired coupler curve to all the curve groups (classes). If any of the curve groups response is "it is a curve within my group", then this group is said to be a "match" with the user-specified curve.

A flow control diagram which depicts the computational model, described above, for the recognition and learning of mechanism coupler curves, is shown in Figure 5.6.

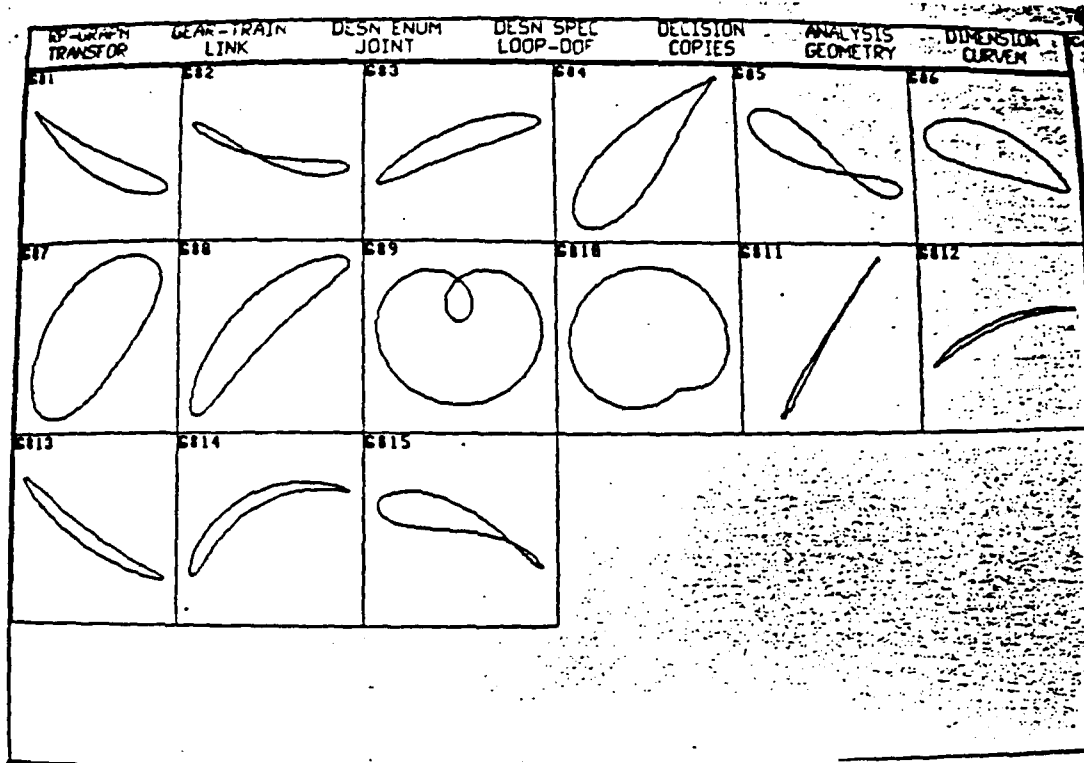


Figure 5.5 A. Classification of 356 coupler curves into 15 groups.

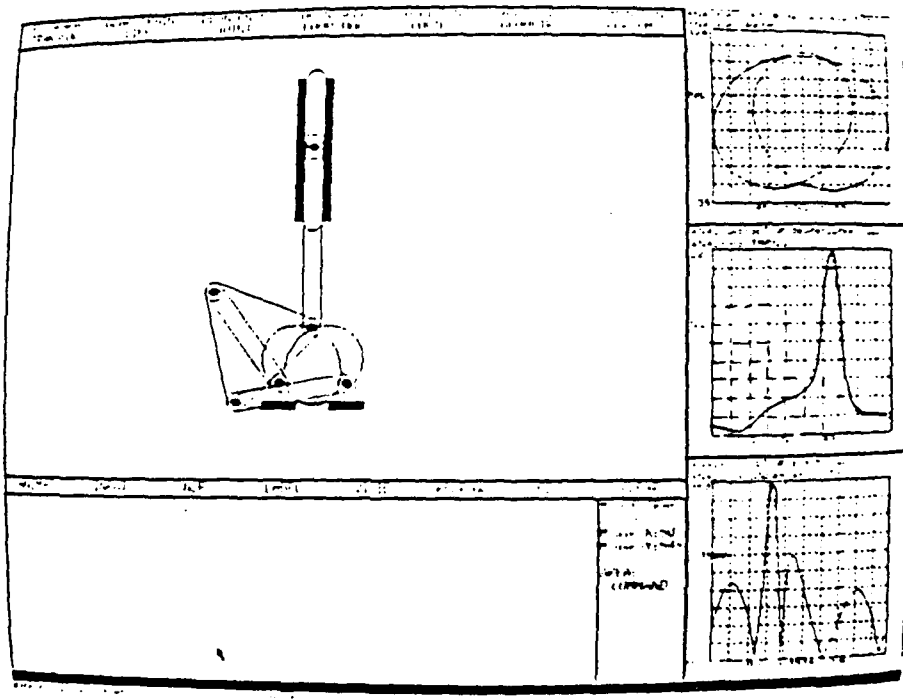


Figure 5.5 B. An example demonstrating the use of Pattern Matching Synthesis for the design of a double-stroke engine or pump mechanism

A. Model

B. Prescriptions:

High level mechanism design language

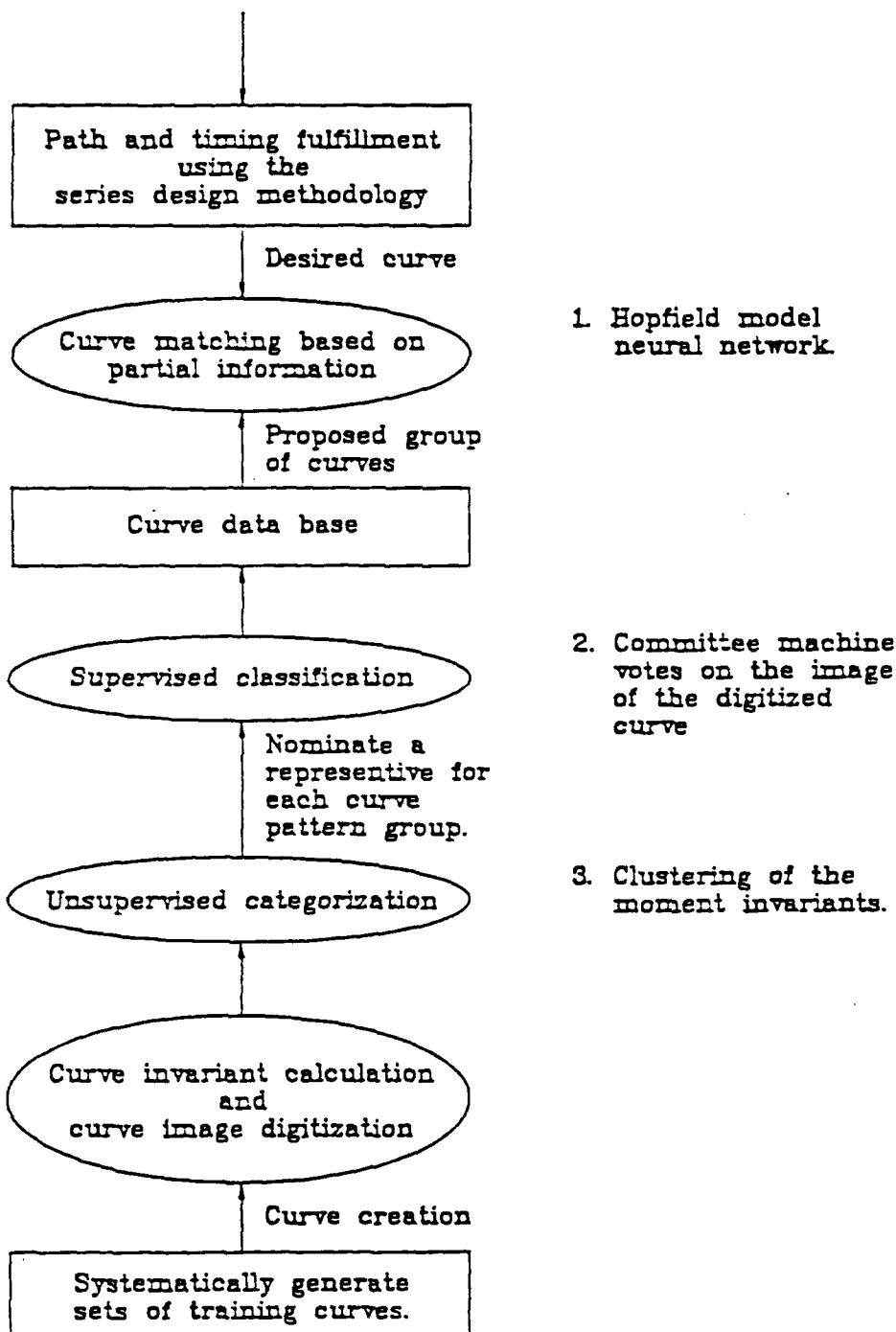


Figure 5.6 A computational model for coupler curve recognition and learning.

6. Kinematic Analysis Based on Object Oriented Programming

6.1 Introduction to Object Oriented Programming.

6.2 Automatic Kinematic Analysis using Object Oriented Programming.

6.3 Automatic Derivation of Closed Form Kinematic Solutions.

6.1 Introduction to Object-Based Programming

Object oriented languages are built around high-level abstractions, and are aimed at simplifying program creation. Object oriented programming simply gives the ownership of procedural constructs to declarative elements, where each element is treated as an object. For example, to check the kinematic constraint of a joint, say joint 2, a traditional program might call a function of the form: *Kinematic-Constraint* (2, joint-position-x-array, joint-position-y-array, joint-type-array, connectivity-matrix, kinematic-constraint-array, 21, 22), while an object-oriented program would ask the *object*, joint-2-obj, to get the kinematic constraint (send joint-2-object :kinematic-constraint), where the kinematic constraint is treated as an explicit "procedure", (to be described in the following sections), whose slot values are instantiated with the link parameters, based on the link adjacency and joint type. The procedures are succinctly shown in Figure 6.1. Object oriented programming languages combine procedures and related data in a single uniform object. This greatly enhances incremental program development and debugging [Cox, '86], [Amir, '89].

Intrinsic properties which maintain the consistency among hierarchically related objects, represents another very important attribute associated with object-oriented programming. Schematic diagrams demonstrating the frame-based and semantic network knowledge representation schemes have been shown in Figure 4.1 A and Figure 4.1 B (Chapter 4).

Consider the following example,

```

Revolute-joint
:is-a (joint)
:instance (joint-1-object, joint-4-object)
:adjacent-to list
:Outer-radius-mm value
:Clearance-mm value

```

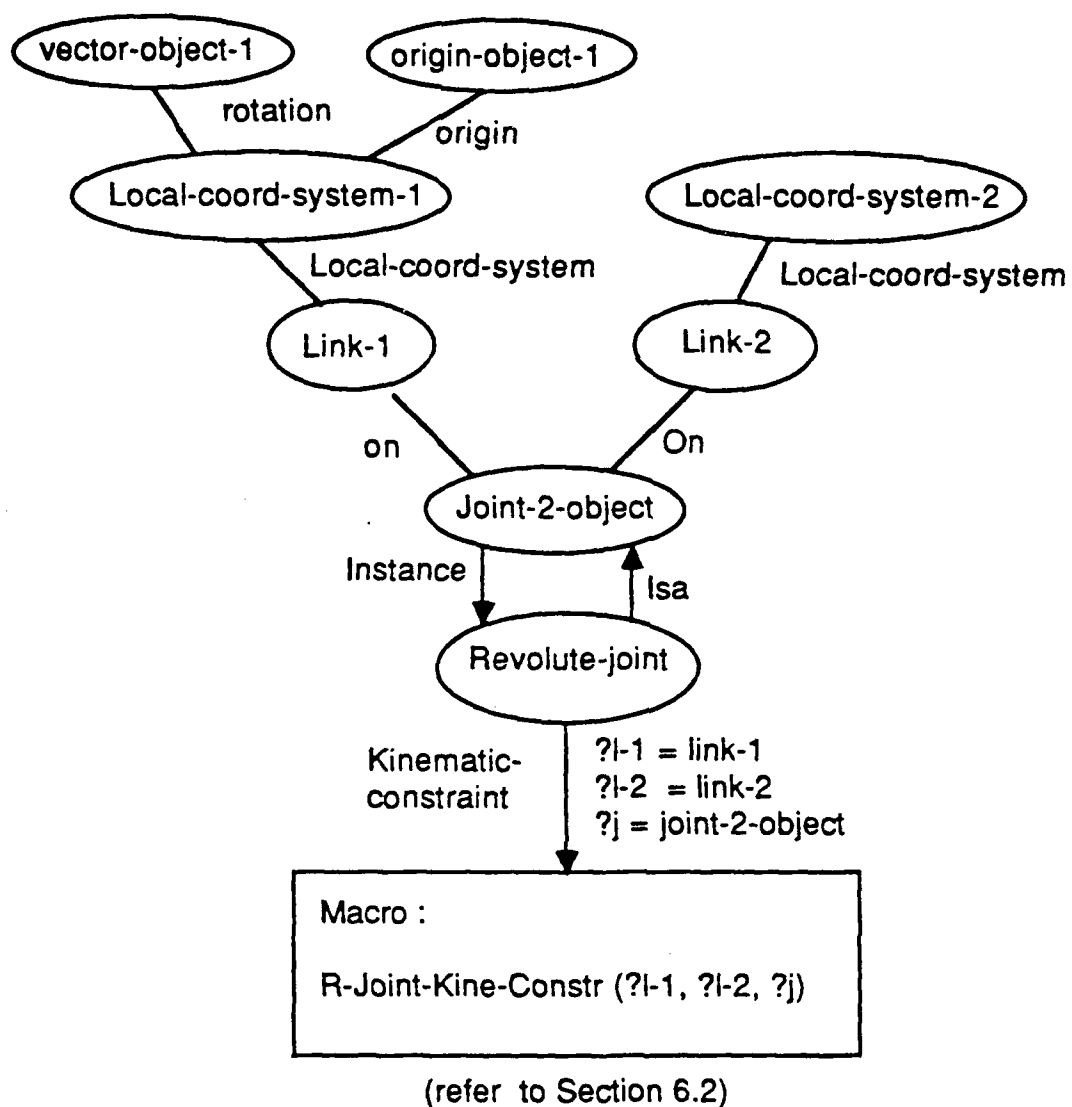



Figure 6.1 Generation of kinematic constraints based on object-oriented programming.

```

:length-mm      value
:degree-of-freedom 1
:contact-type   point
For the creation of a new joint-2-object as a revolute joint,

```

```

Joint-2-object
:is-a (revolute-joint)
:position (:x 10 :y 20 :z 30)
:orientation (:α 0° :β 0° :γ 90°)
:fold 1 {binary, or single joint}
:adjacent-to (joint-1-object joint-4-object)
:Outer-radius-mm 10
:Clearance-mm .01
:length-mm "unknown"

```

Since joint-2-object is a revolute joint, all its default properties, i. e. :isa joint, :degree-of-freedom 1, and :contact-type point, belonging to revolute-joint will automatically be associated with joint-2-object when no further modifications are made to these properties from joint-2-object.

When joint-2-object is created, the relationship established by :adjacent-to is mutual, i.e. if A is adjacent to B, so is B to A, hence the joint-2-object will be added to the slot value of :adjacent-to of joint-4-object and joint-1-object.

Since the inverse relation for :is-a is :instance, the object joint-2-object will also be added to the slot values in :instance of revolute-joint when joint-2-object is created.

When (send joint-2-object :kinematic-constraint) is executed, the position and orientation of joint-2-object will be retrieved as well as the adjacency of joint-2-object. In addition, the constraint equation can be obtained from the joint type relations of the adjacent joints.

6.2 Automatic Kinematic Analysis using Object Oriented Programming

Mechanism animation (calculation of displacements) serves as the basis for all other types of kinematic analysis. Velocity, acceleration, kinetostatic and workspace analysis involve straightforward computations (noniterative).

Mechanism animation must satisfy the kinematic constraints specified by the rigidity of the links and the degrees-of-freedom of the joints. The kinematic constraints for a revolute joint (Figure 6.2), can be expressed as

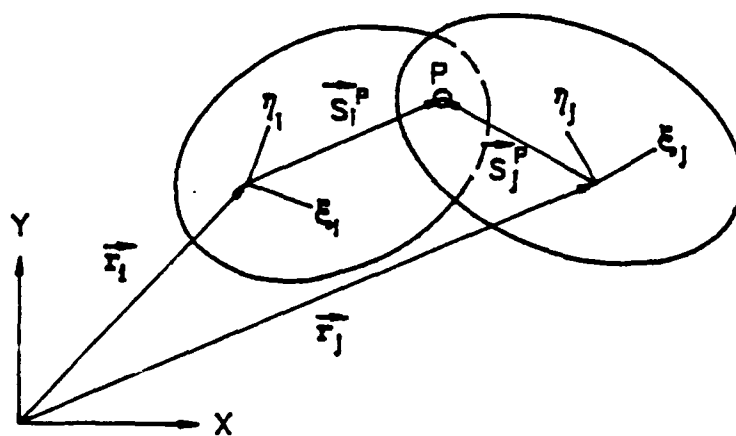


Figure 6.2 An example of a kinematic constraint: a revolute joint p , connects bodies i and j .

follows [Nikravesh, '88].

$$\Phi^{(r,2)} = \vec{r}_i + A_i S_i^p - \vec{r}_j - A_j S_j^p = 0$$

where, (r, 2) means there are two (2), kinematic equations for an "r", revolute joint.

\vec{r}_i is the vector from the global origin to the local origin of body i.

S_i^p is the vector from the local origin of body i to the revolute joint p, in terms of a local coordinate system; S_i^p is a constant vector which represents the rigid body constraint of link i.

A_i is the rotational transformation from the global coordinate system to the local coordinate system.

$\Phi^{(r,2)}$ represents a joint degree-of-freedom constraint based on the rigid body constraint for a link.

The kinematic constraint equations for joints other than revolute joints, that is prismatic, spherical, gear, and other types can also be found in [Nikravesh, '88].

A mechanism animation program based on symbolic computation should be able to obtain all the "active" kinematic constraints, consistent with the general mobility equation. It should also ignore the redundant kinematic constraints which are associated with redundant degrees-of-freedom [Hoeltzel, '86]. An independent kinematic constraint set contains a set of kinematic constraint equations whose number is the same as the number of unknowns included within the equations.

To completely solve the kinematics problem for an entire mechanism with multiple loops, the kinematic constraint equations may be expressed as follows:

$$\begin{bmatrix} [L_1] & [0] & \dots & [0] \\ [0] & [L_2] & \dots & [0] \\ \vdots & \vdots & \ddots & \vdots \\ [0] & [0] & \dots & [L_n] \end{bmatrix} \begin{bmatrix} X \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix}$$

where the $[L_i]$'s are constant coefficient matrices which represents the link and

joint parameters.

The elements of $[X]$ may be nonlinear, but they are partitioned into n independent regions, i.e. $[X] = (X_1 \ X_2 \ .. \ X_n)^T$ where X_i is independent of X_j .

$[L_i] [X_i] = [0]$ is called the i -th independent set.

In order to reduce the complexity of computation, one could individually solve the independent set of kinematic constraint equations instead of simultaneously solving the entire kinematics. From a physical standpoint, this independent set corresponds to the independent loops of the mechanism. This will be explained in the sequel.

6.3 Derivation of a Closed Form Solution for Rapid Kinematic Analysis

The primary bottleneck involved in obtaining a solution to the kinematics problem for animating the motion of a mechanism is the generation of the solution for the nonlinear kinematic constraint equations. Numerical methods for solving a system of nonlinear equations, such as the Newton-Raphson [Nikravesh, '88] method and the path converging method [Morgan, '87], are time consuming and they also tend to be inaccurate due to numerical truncation error. Even worse, they are unable to predict or "foresee" the "locked" positions of the mechanism, i.e., those positions where the mechanism is not able to move. The "locked" positions (those positions where for a specified input position there is no mechanism configuration that will satisfy all the kinematic constraints) cause numerical singularities. Numerically-based solution approaches have to go through (and essentially waste) some number of iterations in order to detect divergence of the kinematic solution.

The loop closure method is typically used for hand derivation of the governing equations for a mechanism kinematics problem. Paul [Paul, '79] has implemented this loop closure method as a computer algorithm. However, the final stage of his method still requires the solution of the nonlinear constraint equations using numerical methods.

Based on an object oriented programming strategy, we have developed a new approach which applies symbolic pattern matching to the independent loops of a mechanism (Figure 6.3). where the solution is prestored in a

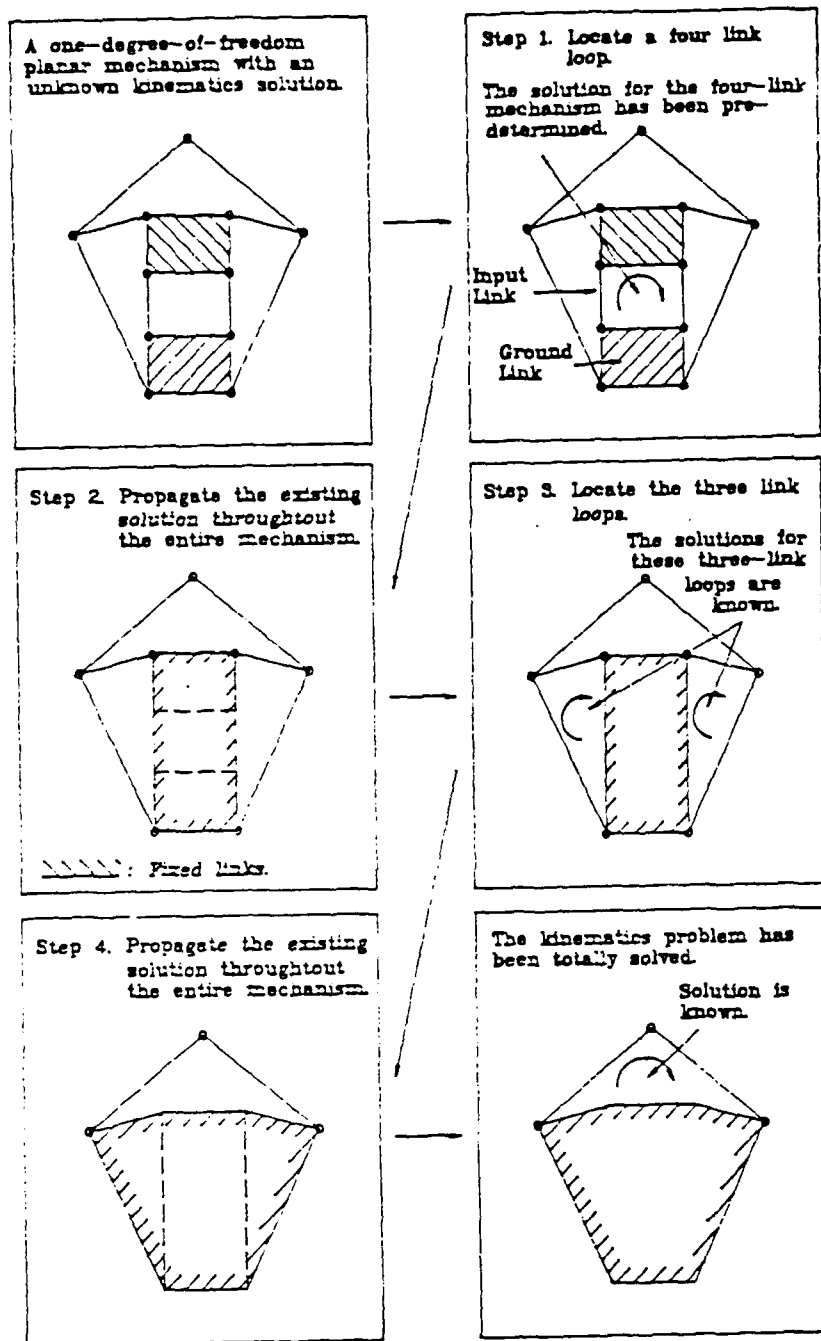


Figure 6.3 An overview of the kinematic loop problem decomposition strategy.

knowledge base. Based on the recognition of kinematic constraint "flow" (an example is shown in Figure 6.4), the kinematic solution for different independent loops can be propagated to other independent loops. Based on this knowledge-based approach, the computer can "derive" the analytical and closed form solutions without the need for numerical calculations.

We have verified this closed form kinematic solution derivation approach [Chieng and Hoeltzel, '88b]. In so doing, we have been able to demonstrate that the kinematic solutions to more than 60% of all ten-link mechanisms, 85% of all eight-link mechanisms and 100% of all six-link and four-link mechanisms can be obtained. As a result of this new approach, we have been able to reduce by 95%, the time required to perform mechanism animation, as compared with traditional numerical methods (Figure 6.5).

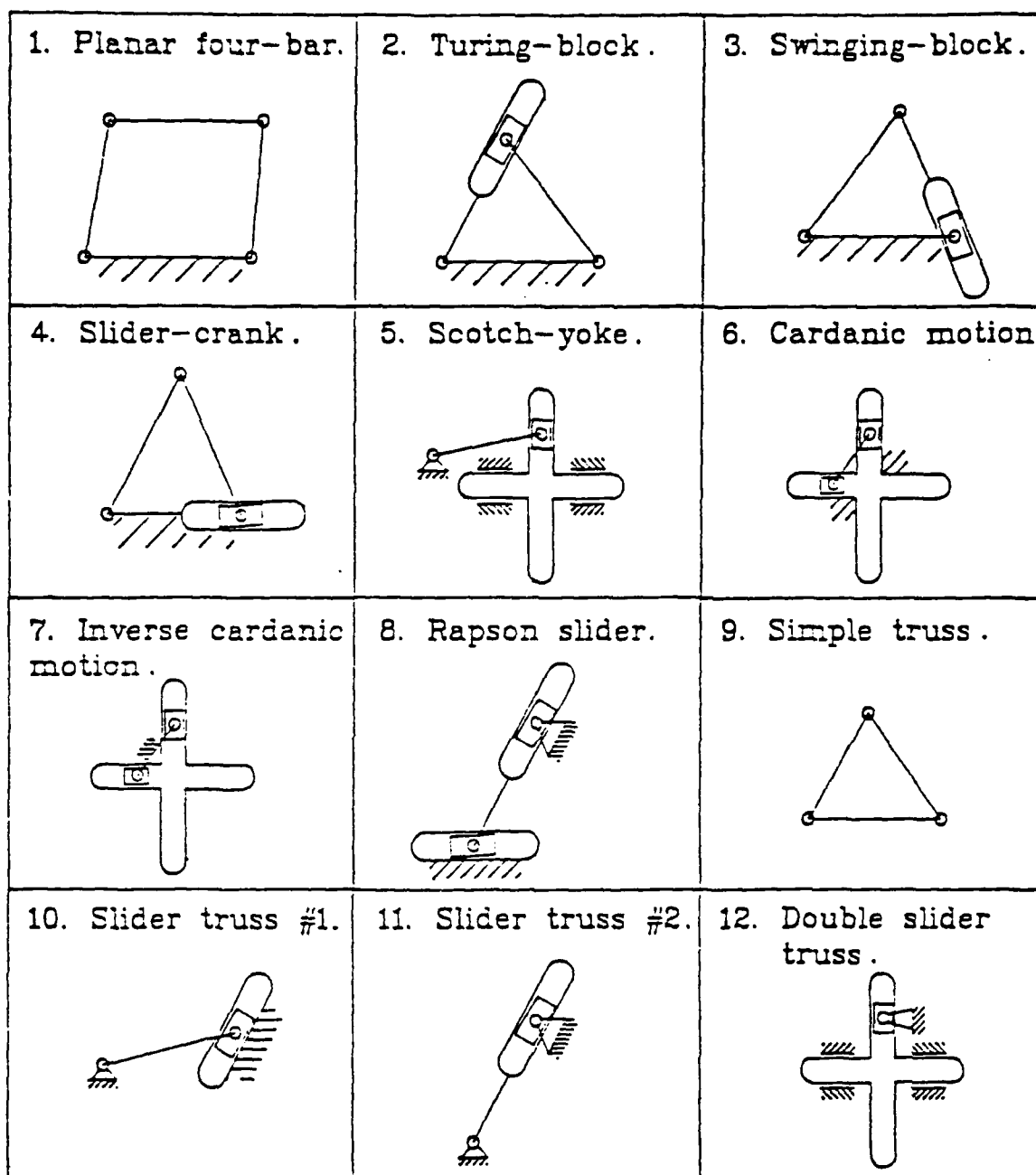


Figure 6.4 Twelve prestored mechanism patterns used for symbolic decomposition of the kinematic problem.

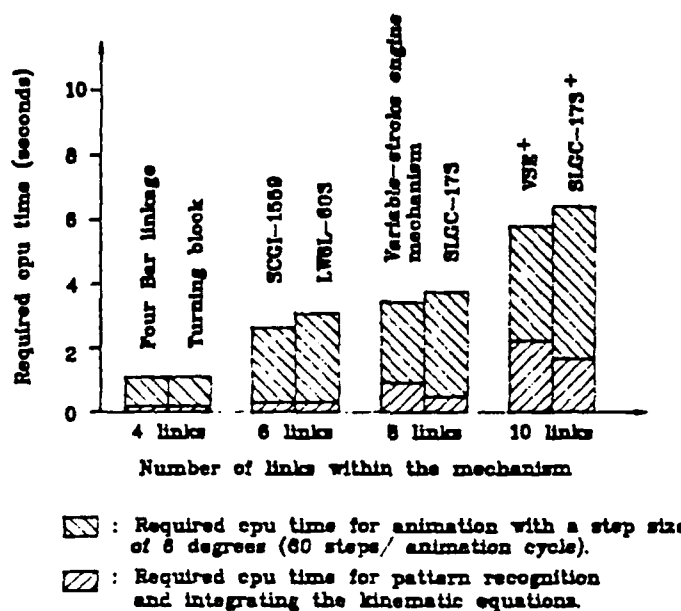


Figure 6.5 A. Algorithmic time complexity as a function of the number of links which comprise the mechanism.

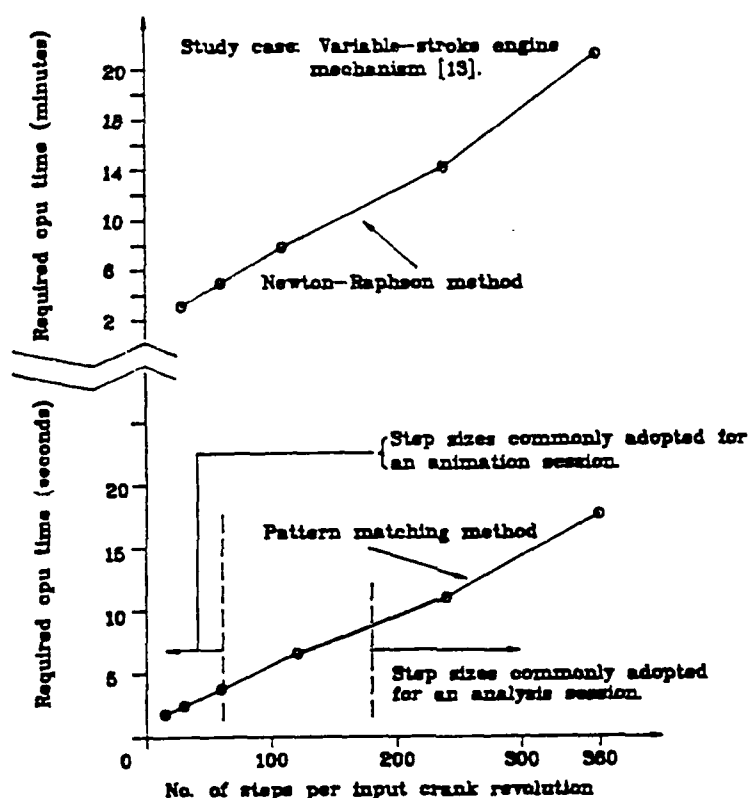


Figure 6.5 B. Real time comparison, performed on Symbolics 3640 computer, between loop decomposition method and the traditional Newton-Raphson (NR) method for solving kinematic equations. (Convergence criteria for NR method are a. positional accuracy = average link length/50 and b. rotational accuracy = .08 radian)

7. Numerical Optimization - Concepts and Applications

7.1 Formulation of a Parametric Optimization Problem.

7.2 Organization of the Mechanism Redesign Process.

7.3 Application to Maximum Mechanical Error Analysis.

7.1 Formulation of a Parametric Optimization Problem

Redesign [Dixon, '86], and design optimization, are usually required when a certain design configuration is unable to satisfy certain design objective metrics. These optimization metrics can include reliability, cost, and performance. In order to achieve an optimal design, minimization or maximization of the design objective requires a search through, and evaluation of various design alternatives. The design constraints serve to bound the search which is driven by the need to improve the design objective; an optimally designed solution must satisfy the design constraints. The mathematical formulation of a parametric design optimization problem may be stated as follows.

Minimize Objective: $f(\mathbf{x})$

where $\mathbf{x} = (x_1, x_2, x_3 \dots x_n)$ is a set of n design variables.

Subject to: inequality constraints: $g_i(\mathbf{x}) \leq 0 \quad i = 1, 2, \dots m.$

equality constraints: $h_j(\mathbf{x}) = 0 \quad j = 1, 2, \dots p.$

Standard nonlinear programming (NLP) methods for which solutions to the parametric optimization problem may be obtained include penalty function methods, reduced gradient methods, sequential linear programming, and sequential quadratic programming, to name a few [Vanderplaats, '84].

Specific nonlinear programming methods each have their own characteristics which enable them to successfully locate a global optimum. Factors which can be used to quantify these characteristics include the degree of nonlinearity and continuity of the functions (objective and constraints), the number of design variables and the number of design constraints, feasibility of the starting design, the cost to evaluate the objective function and others. We have proposed a new approach, based on a concept we refer to as cognitive method switching, for applying nonlinear optimization algorithms in engineering design optimization (Figure 7.1), to gradually and systematically pursue the globally optimized objective

Standard Formulation of a Nonlinear Constrained Optimization Problem:

Minimize:	$F(X)$		objective Function
Subject to:	$G_j(X) < 0$	$j = 1, p$	inequality constraints
	$H_k(X) = 0$	$k = p+1, m$	equality constraints
	$X_l^l < X_i < X_u^u$	$i = 1, n$	variable bounds

Optimization Strategies:

Penalty Function Methods	->	Unconstrained Minimization
Largrange Mutiplier Methods	->	Unconstrained Minimization
Sequential Linear Programming	->	Linear Programming Problem
Sequential Quadratic Method	->	Quadratic Programming (QP) problem
No Strategy		

Search Directions:

For Constrained Minimization:

Feasible Direction Method

Reduced Gradient Method

For Unconstrained Minimization:

Steepest Decent Method

Variable Metric Method, eg, DFP (Davison-Fletcher-Powell)

For Quadratic Programming Problem:

Solution of QP Problem, eg. BFGS (broydon-Fletcher-Goldfarb-Shanno)

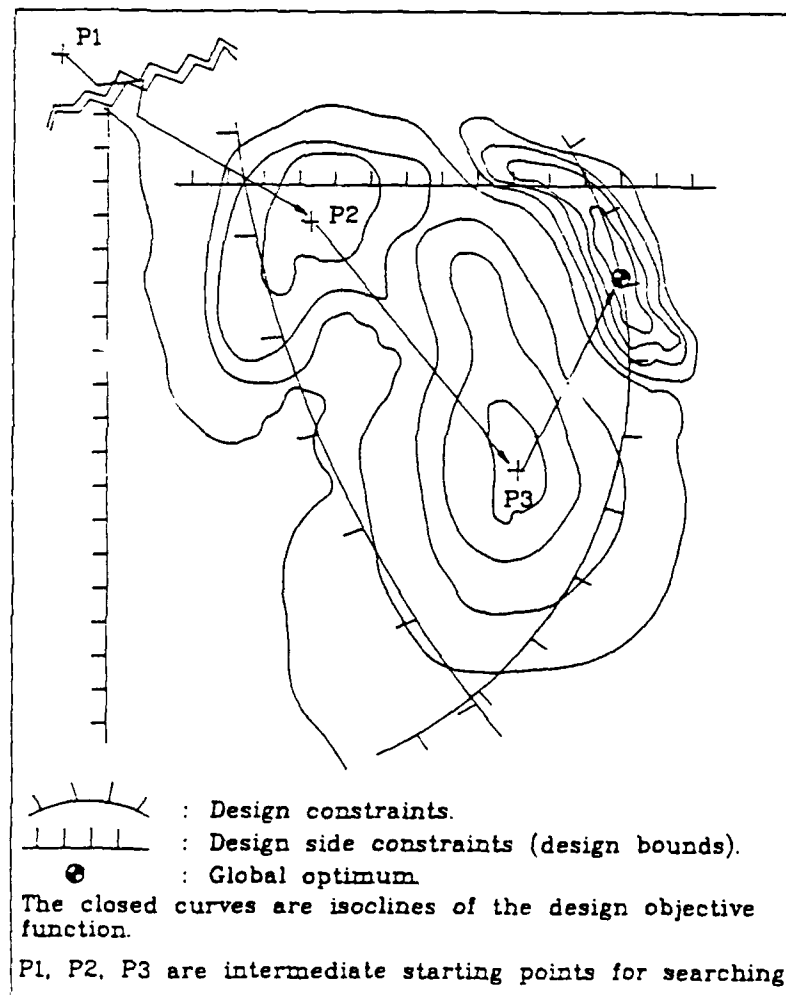


Figure 7.1

function. The basis for cognitive nonlinear programming method switching relies on a statistical machine learning process [Hoeltzel and Chieng, '88b]. Based on knowledge about NLP method selections obtained from the design optimization machine learning process, this program is able to heuristically select the "best" optimization method to satisfy the requirements for different stages of the optimization search.

From the viewpoint of practical mechanical design, discrimination between the design objective and the design constraints is usually unnecessary. For example, the minimization of the capital cost for a product, subject to reliability constraints, is equivalent to a maximization of the reliability subject to a cost constraint. This property is known as mathematical duality [Schjvas, '88]. However, from a mathematical point of view, certain dual problems are simpler than others where numerical design optimization is concerned. Hence, it is quite important to properly pose, from a mathematical standpoint, a design optimization problem.

7.2 Providing Guidance for the Mechanism Redesign Process

To examine the optimality of a design, well known necessary criteria, referred to as the Karush-Kuhn-Tucker (KKT) [Papalambros, '88] conditions, are commonly employed. With respect to the standard formulation of a nonlinear programming problem, as previously described, the KKT conditions can be stated as follows.

If functions f , g and h are continuous and differentiable to the first order, then to minimize $f(\mathbf{x})$ subject to constraints g and h requires:

$$\begin{aligned} \text{KKT conditions: a) } \nabla f + \lambda^T \nabla h + \mu^T \nabla g &= 0^T && \text{The Lagrangian} \\ \text{where } \lambda \geq 0, \mu \geq 0, \mu^T g &= 0. \\ \text{b) } h &= 0, g \leq 0 \end{aligned}$$

The sufficiency condition for a local optimum requires that the Hessian of the Lagrangian be positive definite.

7.3 Application to Maximum Mechanical Error Analysis in Mechanisms

The displacement equation for a mechanism, with k joints and n links, relating the link length parameters $\mathcal{I} = \{l_{ij} \mid i, j \text{ are adjacent joints in the joint adjacency matrix.}\}$ to the respective input and output variables ϕ and ψ ,

may be expressed in general form as follows, where ϕ and ψ may refer to either linear or rotational displacements.

$$\psi = f(\mathcal{L}; \phi)$$

where both ϕ and ψ can be either translational or angular displacement.

The equivalent link lengths [Hoeltzel and Chieng, '89] resulting from joint clearances are denoted as $\mathcal{L}^e = \{l_{ij}^e \mid i, j \text{ are adjacent joints in the joint adjacency matrix.}\}$. For a given value of the input variable ϕ , the corresponding value of the output variable will be $\psi + \Delta\psi$, and may be expressed as follows

$$\psi + \Delta\psi = f(\mathcal{L}^e; \phi)$$

where the mechanical error manifested at the output is denoted by $\Delta\psi$.

To determine the maximum error in the output variable, $\Delta\psi$, due to the joint clearances (the link length tolerances have already been transformed into zero degree-of-freedom joints with joint clearances), the problem can be transformed into the standard form for a nonlinear programming problem, where the objective function is to obtain the maximum $\Delta\psi$ subject to a set of k joint clearance constraints.

Objective function:

$$\text{Maximize } \Delta\psi = f(\mathcal{L}^e; \phi) - f(\mathcal{L}; \phi)$$

$$\text{or Minimize } -\Delta\psi$$

Subject to the inequality constraints:

$$G_j(J_{p,j}, J_{q,j}) \leq \delta_j \quad j = 1, 2, 3 \dots k$$

where link p and link q are connected by joint j .

δ_j denotes the joint clearance for joint j .

$J_{p,j} = (j_x, j_y, j_z, j_\alpha, j_\beta, j_\gamma)_p$ denotes the position and orientation of joint j on link p , and similarly for $J_{q,j}$.

Note: $J_{p,j} = J_{q,j}$ iff there is no joint clearance at joint j .

$G_j(J_{p,j}, J_{q,j})$ is a joint constraint function (indicated in Figure 7.2).

For this problem, if one can find:

1. The solution for l_{ij}^{e*} which induces $G(J_{p,j}, J_{q,j}) = 0$ for all joint j .
2. $\partial\psi^e / \partial l_{ij}^e \big|_{l_{ij}^{e*}} + \lambda^T \nabla G(J_{p,j}, J_{q,j}) = 0$ for all l_{ij}^e .

Joint type	Schematic Diagram	Transient Position And Orientation	Contacting Position and orientation
Planar Revolute Joint		$(X-X_0)^2 + (Y-Y_0)^2 < \delta^2$	$(X-X_0)^2 + (Y-Y_0)^2 = \delta^2$
Planar Prismatic Joint		$ Y-Y_0 < \delta$ $\theta < \tan^{-1}(2\delta/L)$	$Y=Y_0 \pm \delta, \quad \theta = 0^\circ$.OR $\theta = \tan^{-1}(2\delta/L), \quad Y=0.$
Spatial Spherical Joint		$(X-X_0)^2 + (Y-Y_0)^2 + (Z-Z_0)^2 < \delta^2$	$(X-X_0)^2 + (Y-Y_0)^2 + (Z-Z_0)^2 = \delta^2$
Spatial Revolute Joint		$ X-X_0 < \delta_x$ $ Y-Y_0 < \delta_y$ $\theta < \min(\theta_1, \theta_2)$ where $\theta_1 = \tan^{-1}(2\delta_x/L_x)$ $\theta_2 = \tan^{-1}(2\delta_y/L_y)$	$X=X_0 \pm \delta_x,$ $Y=Y_0 \pm \delta_y, \quad \theta = 0^\circ$.OR IF $\theta_1 > \theta_2$ THEN $\theta = \theta_1$ $X=0, \quad Y=Y_0 \pm \delta_y$ ELSE $\theta = \theta_2$ $Y=0, \quad X=X_0 \pm \delta_x$
Spatial Cylindrical Joint		$ Y-Y_0 < \delta$ $\theta < \tan^{-1}(2\delta/L)$	$Y=Y_0 \pm \delta, \quad \theta = 0^\circ$.OR $\theta = \tan^{-1}(2\delta/L), \quad Y=0.$
Linear zero d.o.f Joint (for link Length Tolerance)		$ X-X_0 < \xi$	$X=X_0 \pm \xi$

Figure 7.2 Joint constraints defined in the presence of joint clearances.

3. Hessian of the Lagrangian is positive definite at the local minimum, based on the KKT conditions.

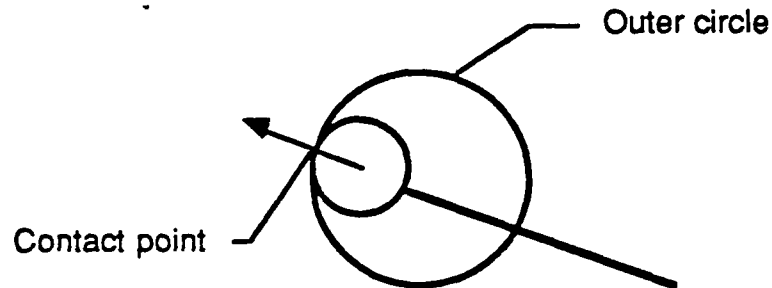
then the maximum mechanical error is guaranteed to exist in the $\mathbf{q}_{ij}^{\epsilon^*}$ configuration. Results which we have presented [Hoeltzel and Chieng, '89], indicate that these conditions can be verified based on the principle of virtual work. Hence, the maximum mechanical error analysis problem can be solved deterministically, rather than stochastically [Dhande, '78].

Four theorems and one corollary, listed in Figure 7.3, have been proven. Based on these four theorems, an algorithm (Figure 7.4), has been developed to determine the maximum mechanical error for general planar and spatial mechanisms. The convergence criteria for each numerical iteration employed in this algorithm is guaranteed by the corollary, when the joint clearance is incrementally added to the analysis.

Figures 7.5, 7.6, and 7.7 demonstrate results based on photos taken from the MEXPERT system as implemented on a Symbolic 3640 AI workstation. The examples show, respectively, the maximum mechanical error for the Schmidt coupling, a parallel-jaw straight line motion generator mechanism, and a sinusoidal function generator mechanism. Each design was fabricated in accordance with different machining processes having various machining tolerance values, as shown in Figure 7.8.

Theorem 1. Subject to the existence of a none-zero external effort, t , the maximum mechanical error for the value of the output variable, j , can only occur while all adjacent links are in contact at a common joint, where there is a none-zero internal effort, i.e., the maximum error occurs when $G_j = \delta_j$.

Theorem 2. For point contact joints, the maximum mechanical error can only decrease with the introduction of friction along joint contacting surfaces. Hence, the contacting point is the point of intersection between the reaction force vector and the outer circle (or sphere in three dimensions) of the joint.



Theorem 3. For line contact joints, the possible contacting position depends on the length of ideal contact, L , the reaction moment, t_j and the reaction force, F_j . If $\tau_j > F_j L/2$, then the joint is in the two point contact position ($w=0$), otherwise it is in the single, side contact line position ($q = 0$).

Theorem 4. The zero degree-of-freedom joint representing link length tolerance must reach its contacting position as shown in Figure 3, when the maximum mechanical error is obtained, unless there is no internal force on the joint.

{When there is no internal force acting on a zero degree-of-freedom joint, this means that there is no stress (either tension or compression) on the associated link.}

Corollary 1. The maximum mechanical error increases monotonically when any of the joint clearances having a non-zero reaction effort is increased.

Figure 7.3 Theorems which support the maximum mechanical error analysis:

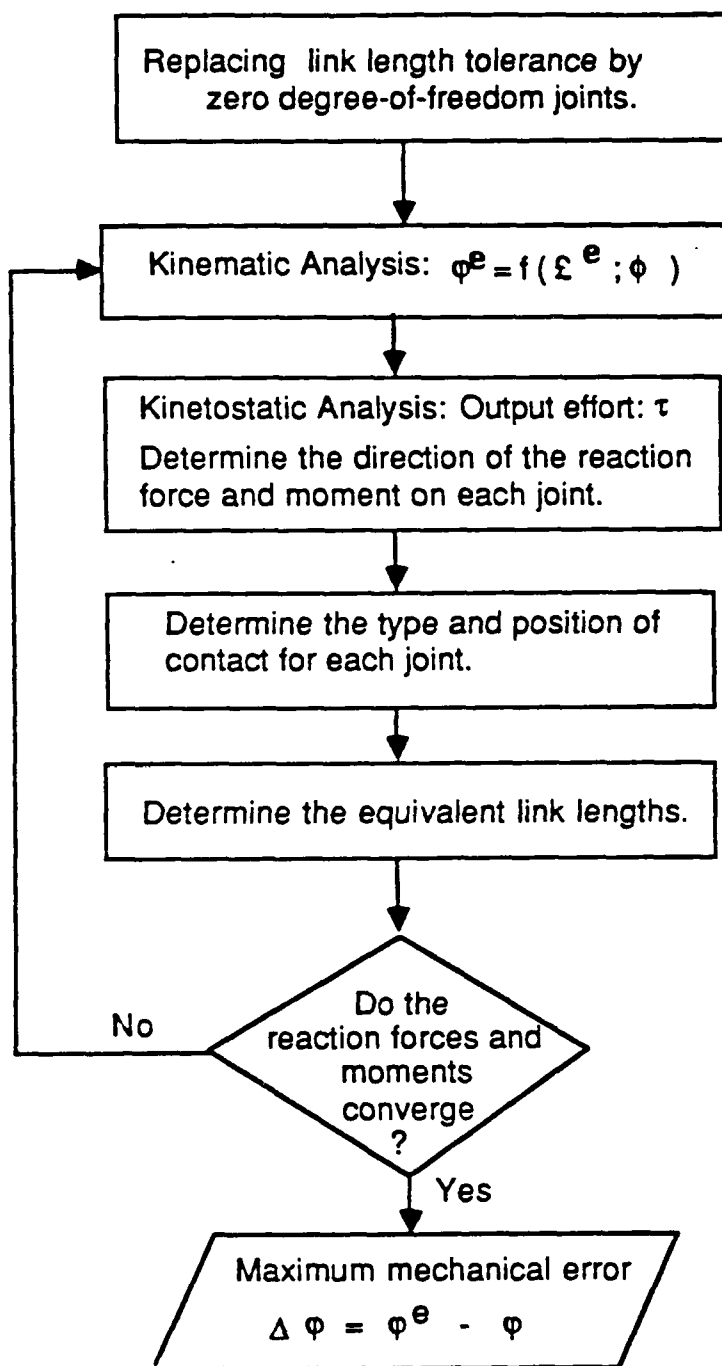


Figure 7.4 Flow control for the maximum mechanical error analysis.

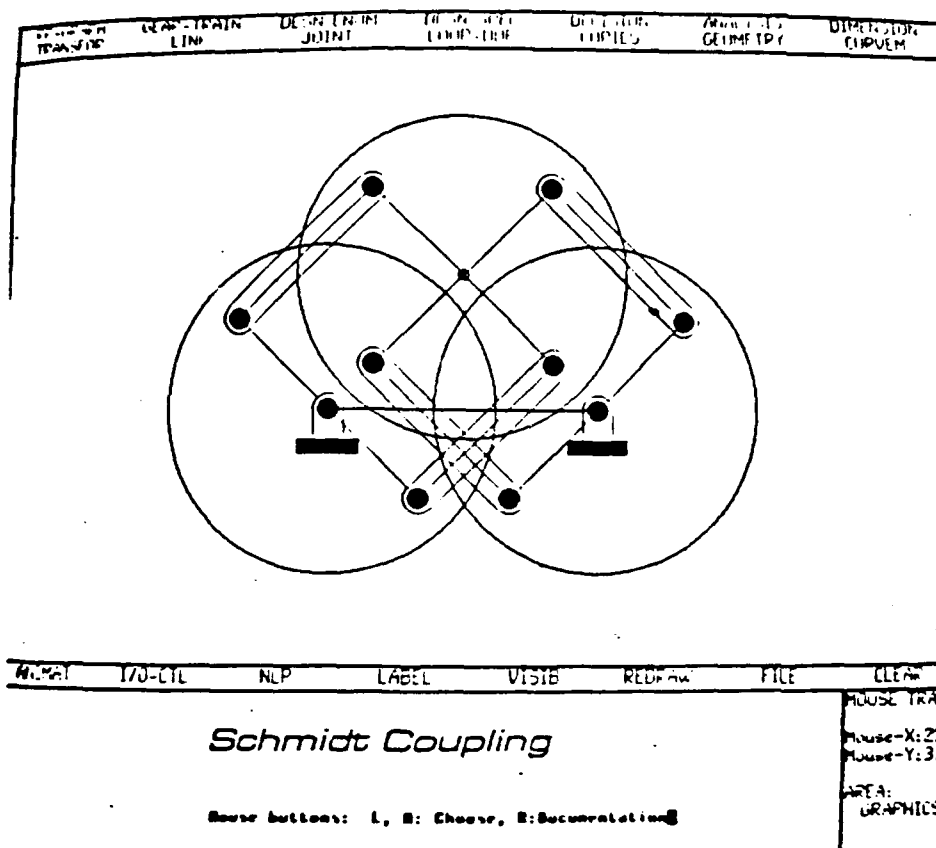


Figure B. Schematic representation of a Schmidt Coupling (front view).

TITLE: JOINT CLEARANCE ANALYSIS			
JOINT NAME	CLEARANCE(.1mm)	RMS SEN(deg/[(0.1x.1mm)])	SEN. INDEX
J1	0.1	0.08	0.12
J2	0.1	0.08	0.12
J3	0.1	0.08	0.12
J4	0.1	0.08	0.12
J5	0.1	0.08	0.12
J6	0.1	0.08	0.12
J7	0.1	0.0	0.0
J8	0.1	0.08	0.12
J9	0.1	0.08	0.12
J10	0.1	0.0	0.0

TITLE: LINK LENGTH TOLERANCE ANALYSIS			
LINK NAME	TOLERANCE(.1mm)	RMS SEN(deg/[(0.1x.1mm)])	SEN. INDEX
0:11-1	0.2	0.0	0.0
0:11-2	0.2	0.07	0.11
0:11-3	0.2	0.08	0.12
0:11-4	0.2	0.08	0.12
0:11-5	0.2	0.14	0.22
0:11-6	0.2	0.08	0.12
0:11-7	0.2	0.08	0.12
0:11-8	0.2	0.07	0.11

Figure D. Table of RMS (root means square) error sensitivities due to joint clearances and link length tolerances.

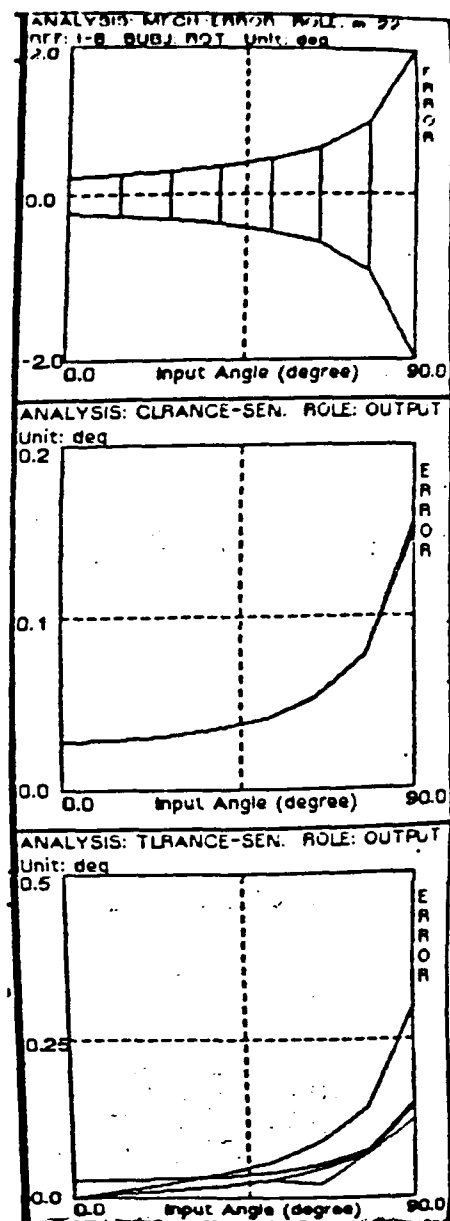


Figure C.
Schmidt coupling
Top: The maximum error produced by joint clearances (0.01mm) and link length tolerances (0.02mm).
Mid: Error sensitivity due to joint clearance (deg/.01mm).
Bot: Error sensitivity due to link length tolerance (deg/.01mm).

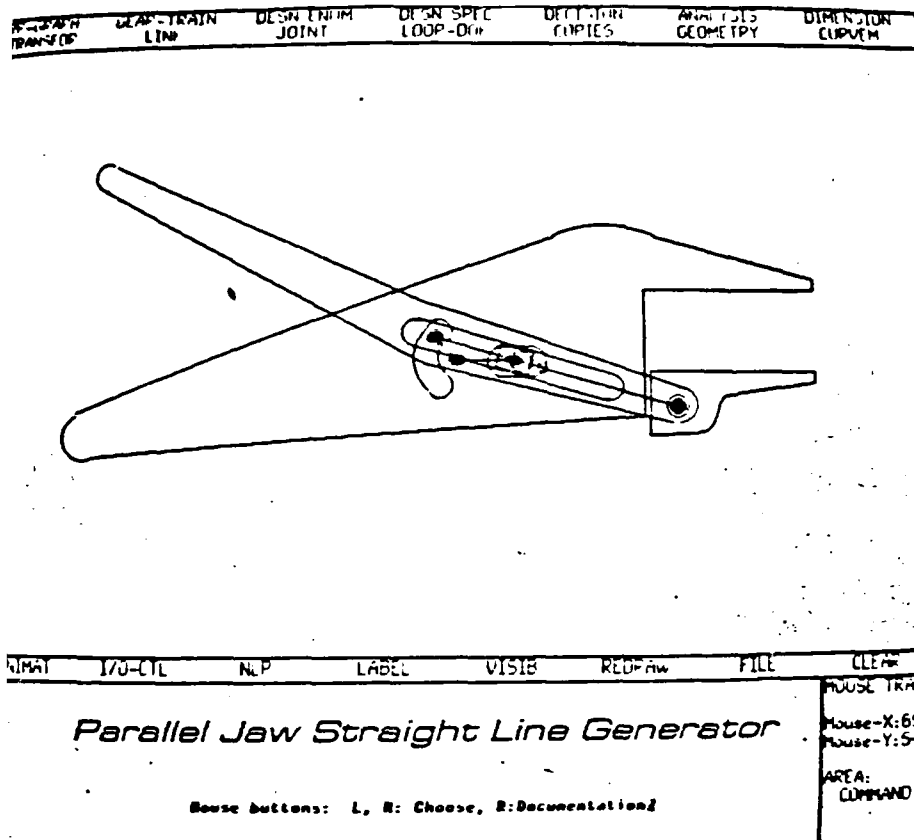


Figure A. Schematic representation of a Parallel-jaw plier.

TITLE: JOINT CLEARANCE ANALYSIS

JOINT NAME	CLEARANCE(.1mm)	RMS SEM(.1mm/[1x.1mm])	SEM. INDEX
J1	0.6	0.83	0.83
J2	0.6	0.83	0.83
J4	0.6	0.2	0.17
J3	0.6	0.2	0.17
NIL	NIL	NIL	NIL

TITLE: LINK LENGTH TOLERANCE ANALYSIS

LINK NAME	TOLERANCE(.1mm)	RMS SEM(.1mm/[1x.1mm])	SEM. INDEX
0: [1-1]	0.6	0.54	0.53
0: [1-2]	0.6	0.83	0.83
0: [1-3]	0.6	0.49	0.49
0: [1-4]	0.6	0.1	0.08

Figure C. Table of RMS (root means square) error sensitivities due to joint clearances and link length tolerances.

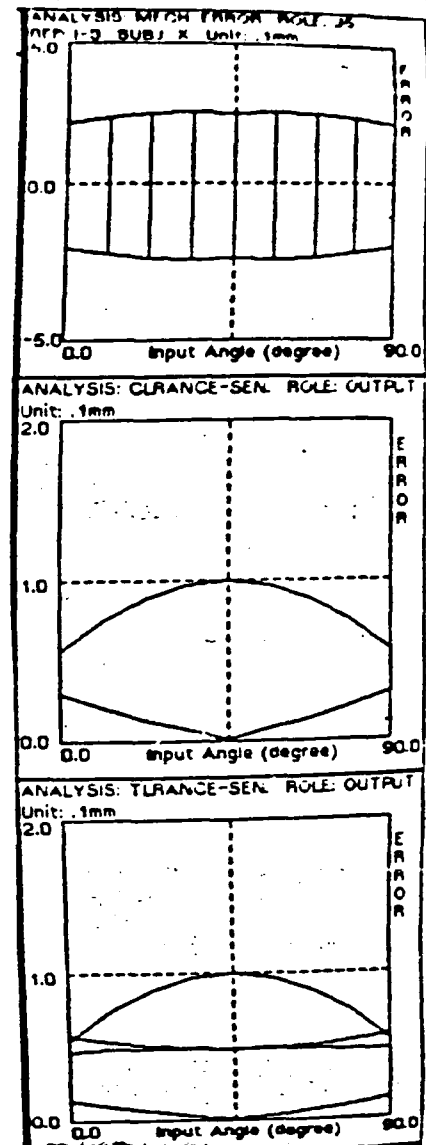


Figure B.
Parallel-jaw plier
Top: The maximum error produced by joint clearances (0.06mm) and link length tolerances (0.06mm).
Mid: Error sensitivity due to joint clearance (.1mm/.01mm).
Bot: Error sensitivity due to link length tolerance (.1mm/.01mm).

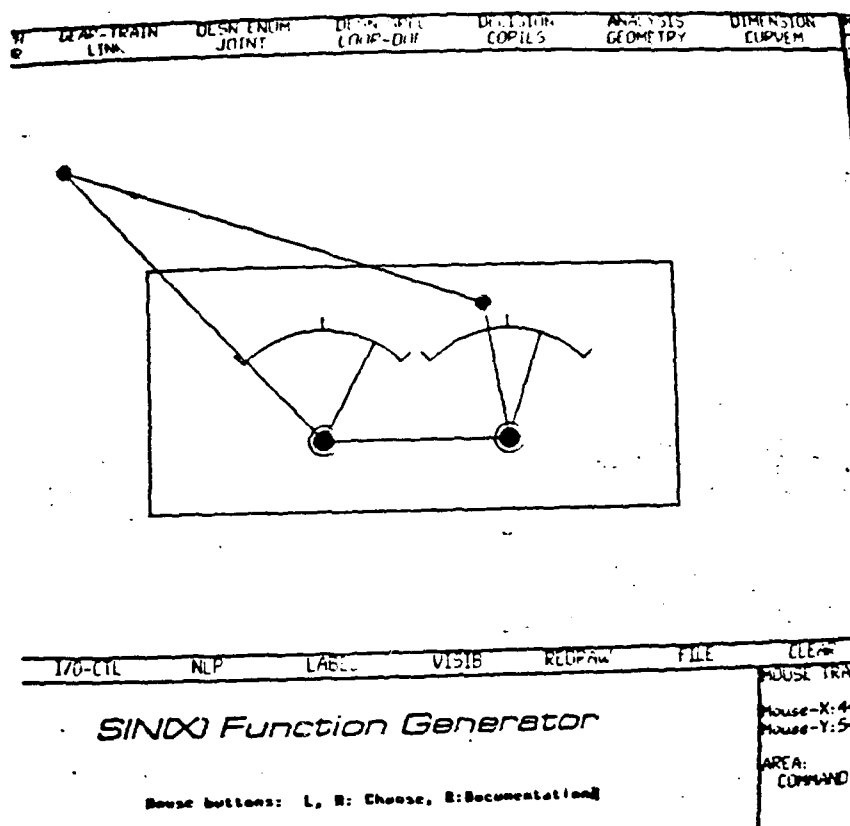


Figure A. Schematic representation of a Sinusoidal function generator.

TITLE: JOINT CLEARANCE ANALYSIS

JOINT NAME	CLEARANCE(.1mm)	RMS SEN(deg/[(8.1x.1mm)])	SEN. INDEX
JA	0.2	0.11	0.11
JD	0.2	0.11	0.11
JB	0.2	0.11	0.11
JC	0.2	0.11	0.11

TITLE: LINK LENGTH TOLERANCE ANALYSIS

LINK NAME	TOLERANCE(.1mm)	RMS SEN(deg/[(8.1x.1mm)])	SEN. INDEX
0:11-11	0.2	0.1	0.11
0:11-21	0.2	0.1	0.11
0:11-31	0.2	0.11	0.11
0:11-41	0.2	0.07	0.08

Figure C. Table of RMS (root means square) error sensitivities due to joint clearances and link length tolerances.

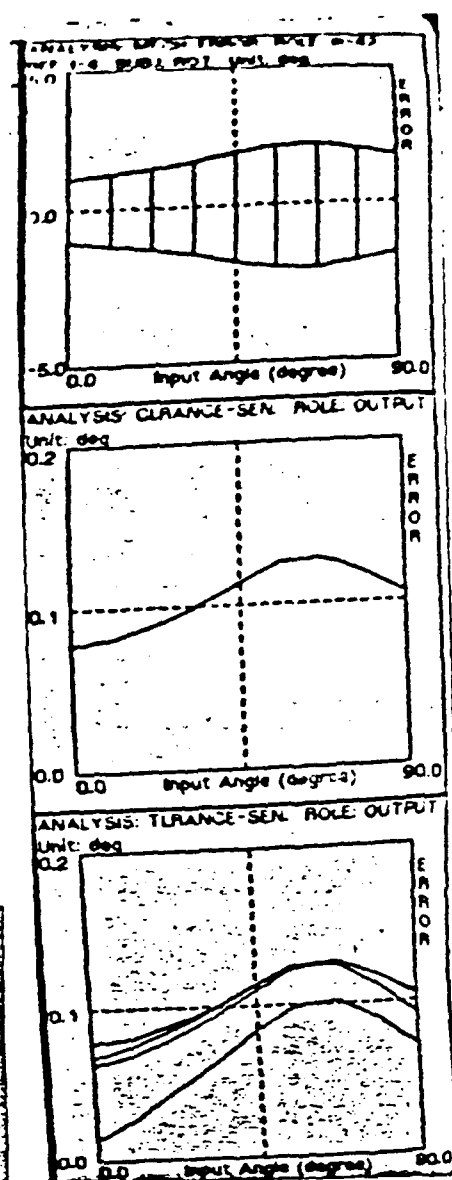








Figure B.
Sinusoidal function generator
Top: The maximum error produced by joint clearances (0.02mm) and link length tolerances (0.02mm).
Mid: Error sensitivity due to joint clearance (.1mm/.01mm).
Bot: Error sensitivity due to link length tolerance (.1mm/.01mm).

Unit: mm

Range of sizes		Manufacturing tolerance									
From	To										
0.00	15.23	.00381	.00508	.00736	.01270	.02030	.03048	.05080	.07620	.1270	
15.24	25.39	.00381	.00638	.01016	.01524	.02540	.03810	.06350	.1016	.1520	
25.4	38.09	.00508	.00736	.01270	.02030	.03048	.05080	.07620	.1270	.2030	
38.1	71.11	.00635	.01016	.01524	.02540	.03810	.06350	.1016	.1520	.2540	
71.12	114.29	.00763	.01270	.02032	.03048	.05080	.07620	.1270	.2030	.3048	
114.3	198.09	.01016	.01524	.02540	.03810	.06350	.1016	.1520	.2540	.3810	
198.1	350.49	.01270	.02032	.03048	.05080	.07620	.1270	.2030	.3048	.5080	
350.5	533.4	.01524	.02540	.03810	.06350	.1016	.1520	.2540	.3810	.6350	

Machining Process	Tolerance range
Lapping & Honing	
Grinding, Diamond Turning & Boring	
Broaching	
Turning, Boring, Slotting, Planing & Shaping	
Milling	
Drilling	

(adapted from Giesecke, F.E., Mitchell, A., Spencer, H.C., and Hill, I.L. (1967):
Technical Drawing, The Macmillan Co., 5th Edition, pp. 356)

Figure 7.8 Tolerance ranges for various manufacturing processes.

8. The Hybrid (Sybolic - Numeric) Optimization Methodology

8.1 The Hybrid Optimization Methodology.

8.2 Hybrid Optimization Applied to the Design of a Universal Joint.

8.1 A Systematic Methodology for Hybrid Optimization

A Hybrid (Symbolic-Numerical) expert system is a software-based (typically) system which allows for an interconnection to be established between design experience (i.e., design heuristics) and traditional numerical computing [Kitzmillen, et al., '87].

A hybrid design system model [Chieng and Hoeltzel, '87] separates the design optimization processes into two design levels, a level for conceptual design synthesis and one for detailed design analysis. In the conceptual design stage, the primary design decision making processes are performed symbolically, while during the detailed design stage the primary decision making processes are performed numerically. The hybrid design methodology is applicable to design situations where a clear separation exists between the symbolic and numerical design processes. For example, this methodology clearly applies to mechanism synthesis and analysis. For problems where a clear separation does not exist, it may be appropriate to couple the stages together and to perform the entire design optimization process in a more integrated fashion. Various architectures which combine symbolic and numerical processes together are described in [KitzMiller, '87].

In developing a hybrid optimization system (Figure 8.1), which is capable of manipulating symbolic knowledge and performing optimization on symbolic knowledge, an expert system approach is sought. In accordance with the design heuristics obtained from a real world, engineering designer, optimization must be performed heuristically, and therefore, a global optimum cannot be guaranteed. We therefore refer to this as "near optimal design". During the detailed design stage, optimization is performed by nonlinear programming algorithms, or by linear programming algorithms, if the governing equations are linear. A design equation data base has been established which contains all the necessary design equations. This data base is generated either from emperical data or from theoretical considerations. Pertinent design equations are usually available in design

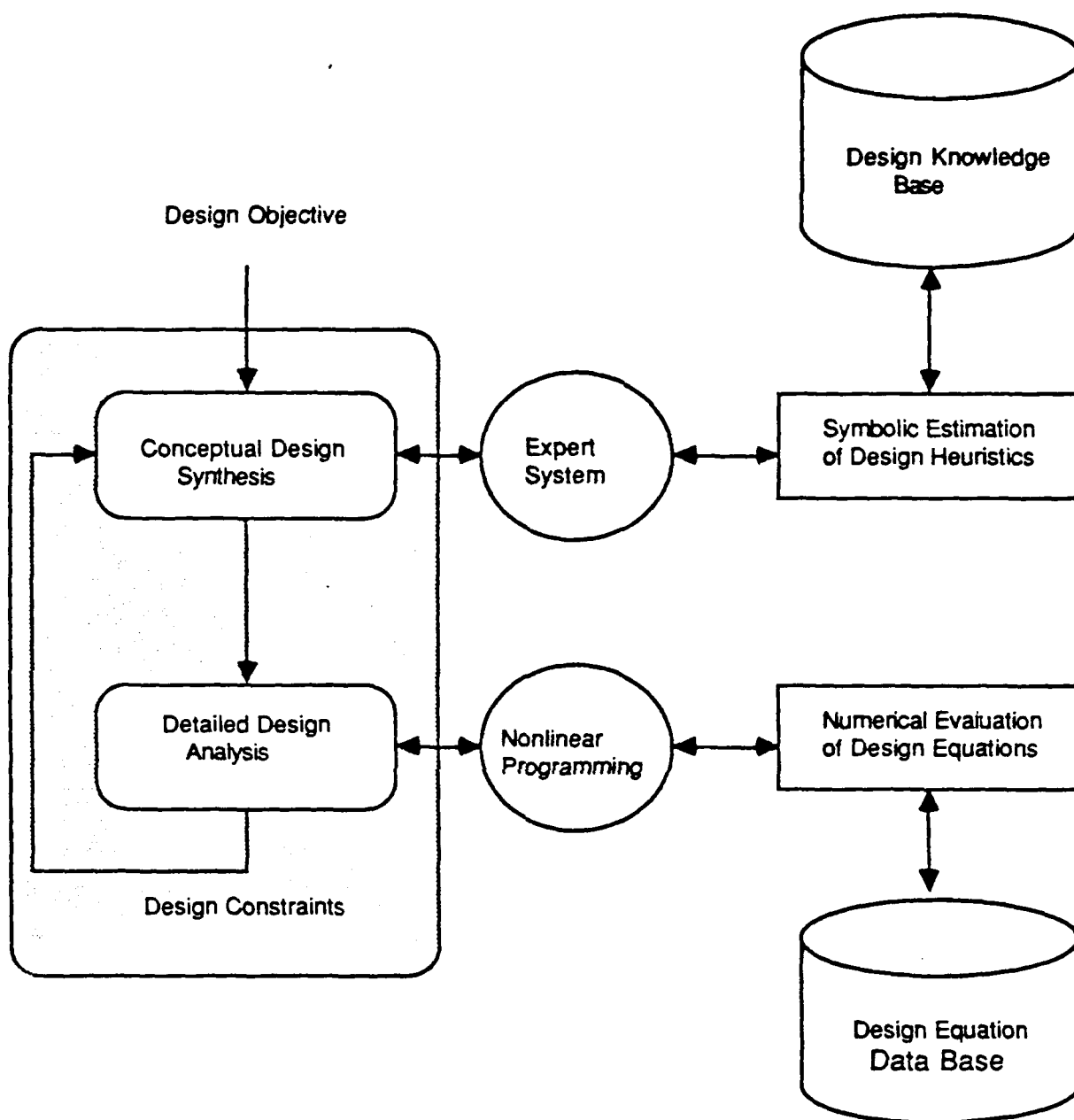


Figure 8.1 A model for hybrid optimization.

handbooks or from appropriate ASME design codes [Shigley, '77].

8.2 Hybrid Optimization Applied to the Design of a Universal Joint

The traditional (cross and yoke) universal joint [Fischer, '84] contains redundant constraints which have the potential to a) reduce the reliability and b) increase the difficulty required to successfully assemble the universal joint, subject to certain manufacturing tolerances. As shown in Figure 8.2, the traditional cross and yoke-type universal joint contains seven redundant constraints.

Based on these facts, there is a reason to redesign the universal joint such that the redundant constraints are eliminated. A hybrid design process to redesign a universal joint follows.

Symbolic optimization - Kinematic structure optimization:

- a. Degree-of-freedom assignment: Based on heuristic knowledge about degree-of-freedom assignment, the degrees-of-freedom are distributed to each of the intermediate joints, while the sum of the degrees-of-freedom must be equal to 19. This process is shown in Figure 8.3a.
- b. Joint type selection for each intermediate joint: Based on e knowledge about joint configurations, as shown in Figure 8.4b, each joint type has been searched and evaluated. This is done for each intermediate joint. The heuristic knowledge required for the selection of different joint types is stored in a knowledge base.

Following the joint degree-of-freedom assignment and joint type selection processes, an optimal kinematic structure for the universal joint can be determined (Figure 8.4a). We refer to this redesigned configuration as a Ball-pin type universal joint.

Numerical optimization - Physical dimension optimization.

A physical representation for Ball-pin type universal joint has been demonstrated in Figure 8.5a and 8.5b. Subject to design constraints on bending stress, yielding stress, surface fatigue and to geometrical constraints, the objective is to minimize the size of the universal joint when different maximum load levels (torques) are applied. Following the nonlinear programming optimization process, a minimally

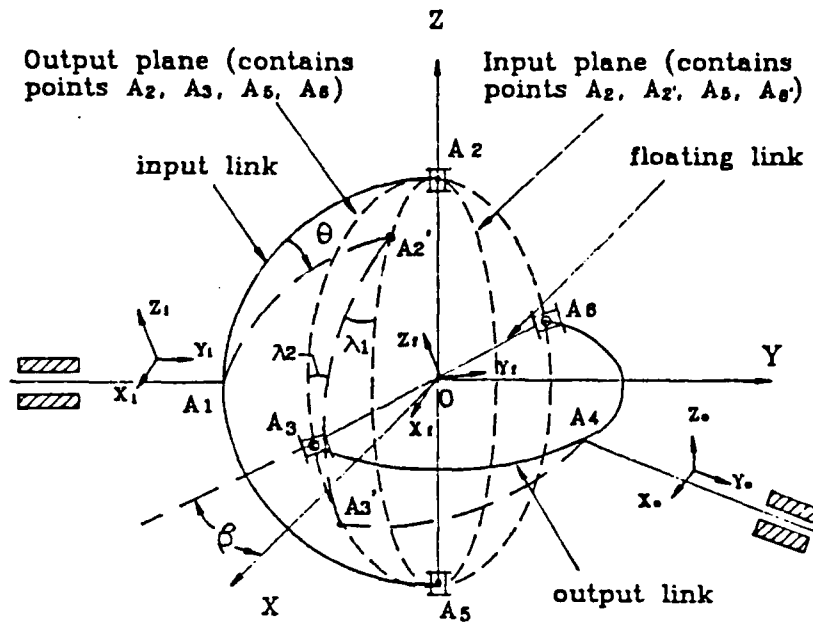


Figure 8 2 A Schematic diagram of a cross-and-yoke type universal joint.

Kinematic factors	Cross and yoke U-joint	Ball-pin U-joint
F (number of dof)	1	1
l (number of links)	4	8
j (number of joints)	6	10
A (total number of dof of all the joints)	12	19
λ (dof of space)	6	6
q (number of redundant constraints, Eq. 4b.)	7	0

Figure 8.2 B Kinematic comparison between traditional cross-and-yoke type universal joint and ball-pin type universal joint.

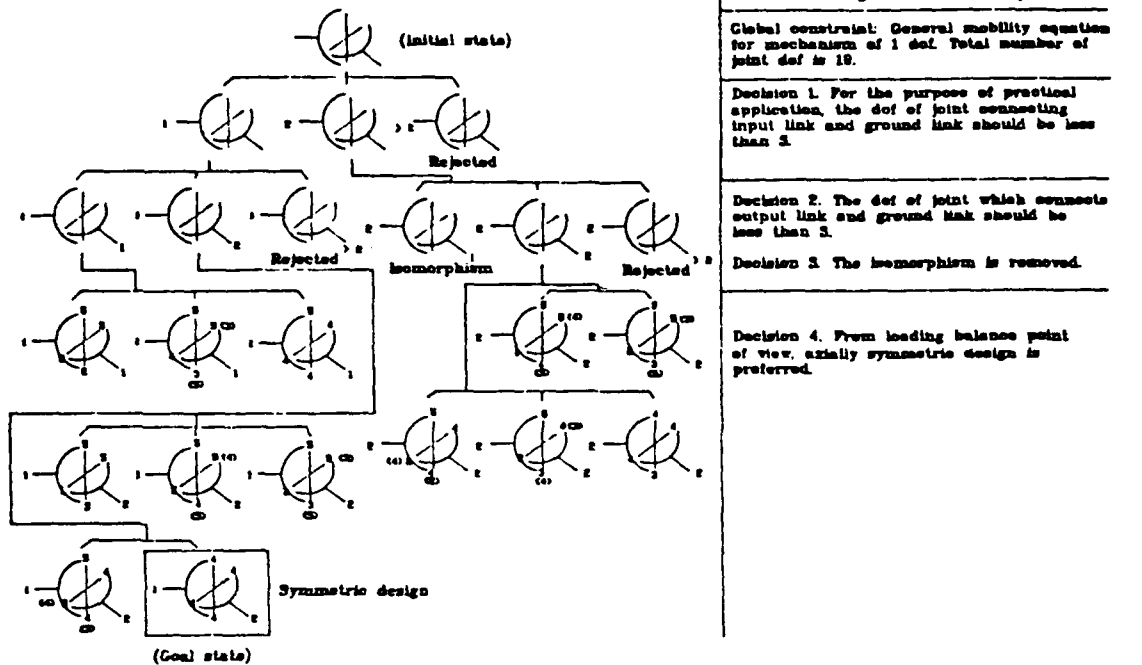


Figure 8.3 A Intermediate assignment of joint degrees-of-freedom.

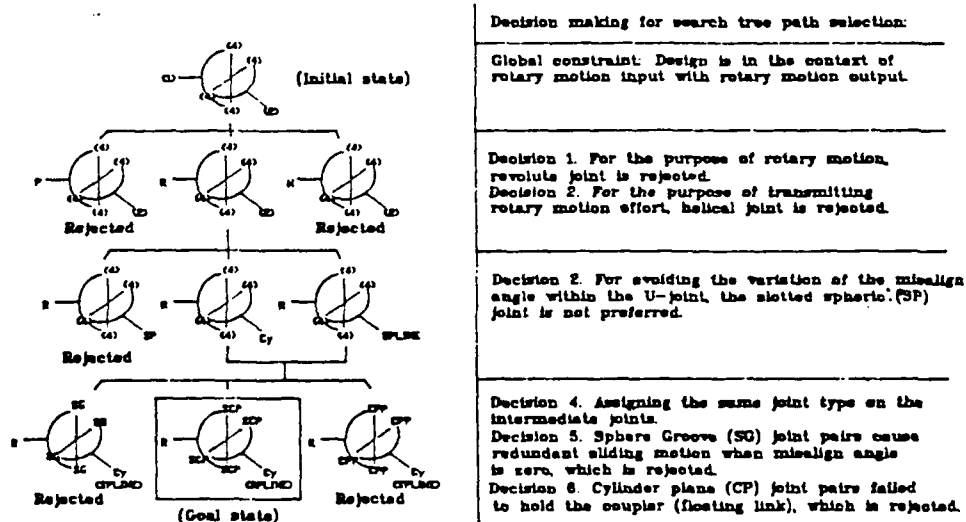
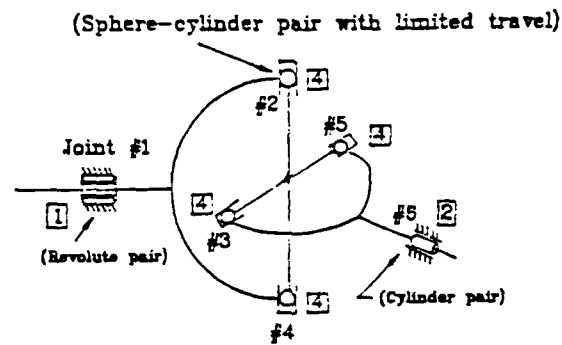


Figure 8.3 B Intermediate selection of joint types.



1: number of degrees-of-freedom per joint

Figure 8.4 A Optimal kinematic structure for the cardanic joint (universal joint).

Total degrees-of- freedom	D. O. F. A.	Theoretical higher pair	Combination of lower pairs	D. O. F. A.	Theoretical higher pair	Combination of lower pairs	D. O. F. A.	Theoretical higher pair	Combination of lower pairs
5	T^2 R^3			T^3 R^2	Does not exist				
4	T^1 R^3			T^2 R^2 (I)			T^2 R^2 (II)		
	T^2 R^2 (III)	Does not exist		T^3 R^1	Does not exist				
3	R^3			T^1 R^2 (I)			T^1 R^2 (II)	Does not exist	
	T^2 R^1 (I)			T^2 R^1 (II)	Does not exist		T^3	Does not exist	
D.O.F.A: degree-of-freedom assignment T: translation R: rotation									

Figure 8.4 B Joint design: Combining lower pairs to produce kinematically equivalent higher pairs.

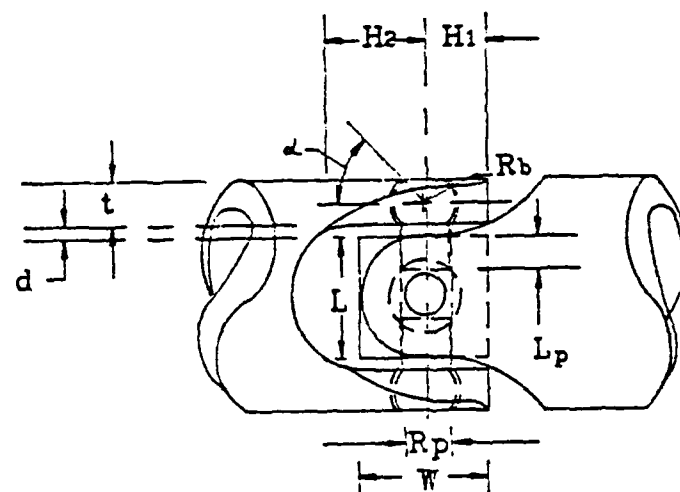


Figure 8.5 A Physical representation of a Ball-pin type universal joint.

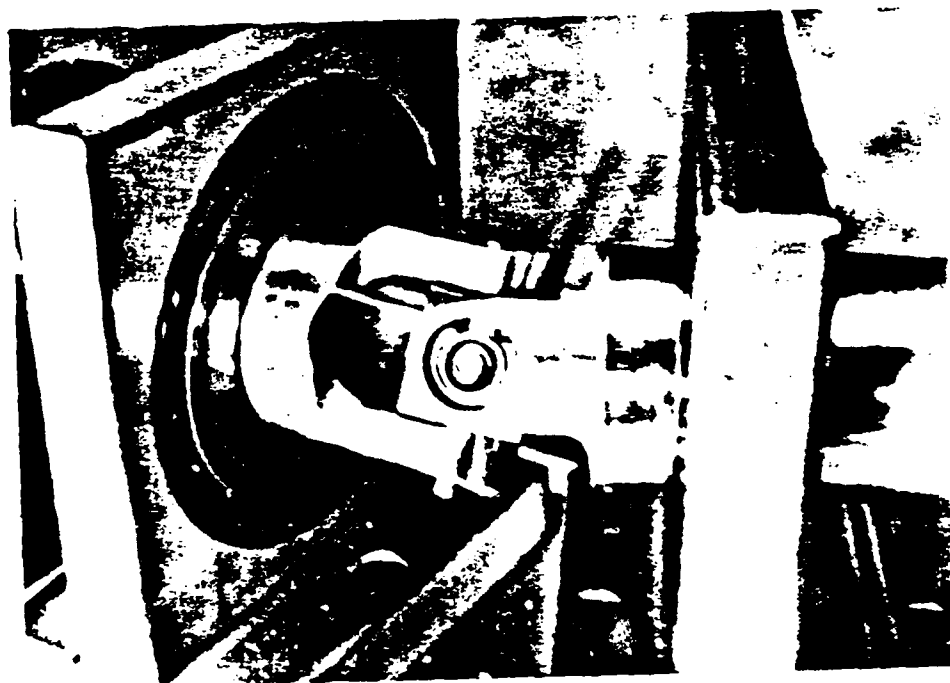
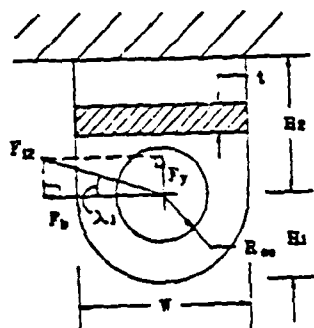
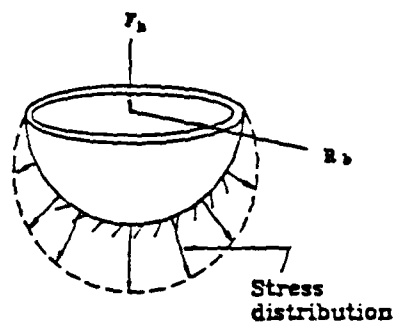


Figure 8.5 B Photograph of a Ball-pin universal joint.
(Columbia Univ., Mechanical Eng. Dept, Machine Shop)

sized Ball-pin type universal joint can be obtained (Figure 8.6a), and the corresponding optimal (mimimal) dimensions for the Ball-pin type universal joint can be determined.



(a)



(b)

Bending fatigue due to reaction forces:

$$(\sigma_b)_{\max} \geq 12 \cdot K_b \cdot F_b \cdot H_1 \cdot R_{ec} / (W - 2R_{ec})^3 t$$

Yielding fatigue due to reaction forces:

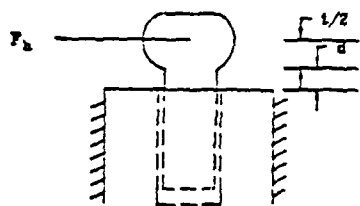
$$K_y = 7.778 - 18.6 \eta + 23.26 \eta^2 - 15.6 \eta^3$$

$$(\sigma_y)_{\max} \geq K_y \cdot F_y / (W - 2R_{ec}) t$$

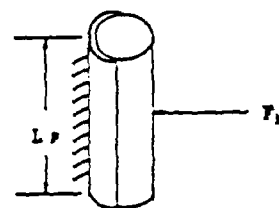
the surface fatigue constraint is,

$$(\sigma_{rs})_{\max} = 2F_h / \pi R_b^2 (\alpha - \sin 2\alpha / 2)$$

$$(\sigma_s)_{\max} \geq (\sigma_{rs})_{\max}$$



(c)



(d)

Bending fatigue due to the moment :

$$M_b = F_h \cdot (t/2 + d)$$

$$K_{b1} = 2.3$$

$$(\sigma_b)_{\max} \geq 32 \cdot K_{b1} \cdot F_h / \pi R^2$$

The surface contact fatigue constraint of the floating block is,

$$(\sigma_s)_{\max} \geq (\sigma_{rc})_{\max} = 2P / \pi R_p$$

Geometrical constraints:

$$H_2 \geq R + t/2$$

Figure 8.6 Design constraints for dimensioning the Ball-pin type universal joint.

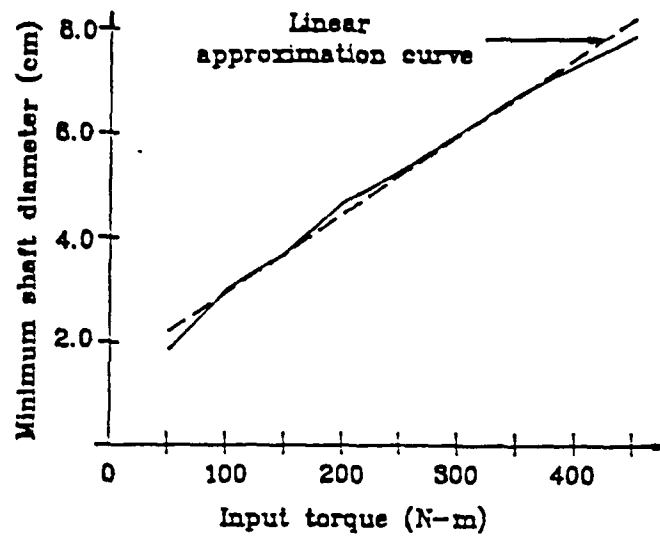


Figure 8.7 A Minimum shaft diameter vs. input torque.

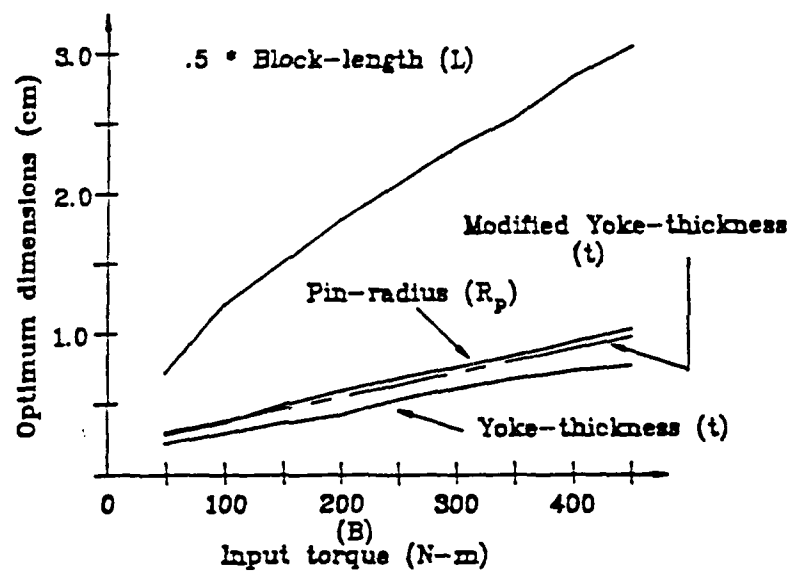


Figure 8.7 B Optimal dimensioning of Ball-pin type universal joint.

sized Ball-pin type universal joint can be obtained (Figure 8.6), and the corresponding optimal (mimimal) dimensions for the Ball-pin type universal joint can be determined (Figure 8.7).

References

- Aho, A.V., Hopcroft, J.E., Ullman, J.D. (1983): *Data Structures and Algorithms*, Addison-Wesley Pub. Co., Reading, Massachusetts.
- Ambekar, A.G. and Agrawal, V.P. (1987): "Canonical Numbering of Kinematic Chains and Isomorphism Problem: Min Code" in *J. of Mechanism and Machine Theory*, Vol 22, No. 5, pp. 453-461.
- Amir, S. (1989): "Building Integrated Expert Systems" in *AI Expert Magazine*, January 1989.
- Atkinson, A.O. (1985): "Design for Manufacturability: Computer-Integrated Design and Manufacturing for Product Development" SAE Technical Paper Series No. 851587.
- Bow, S.T. (1984): *Pattern Recognition - Applications to Large Data-Set Problems*, Marcel Dekker Inc, New York, pp. 18 -65.
- Brown, D.C. and Chandrasekaran, B. (1986): "Knowledge and Control for a Mechanical Design Expert System", *IEEE Computer*, Vol. 19, pp. 92-100.
- Charniak, E. and Modermott, D. (1985): *Introduction to Artificial Intelligence*, Addison-Wesley Pub. Co., Reading, Massachusetts.
- Chieng, W.H. and Hoeltzel, D.A. (1986): "Hybrid (Symbolic-Numeric) Optimization of Universal Joint Design with Application to the Ball-pin Type Universal Joint.", ASME paper No. 86-DET-123.
- Chieng, W.H. and Hoeltzel, D.A. (1987): "Interactive Hybrid (Symbolic - Numeric) System Approach to Near Optimal Design of Mechanical Components" in *Engineering with Computers*, Vol. 2, No. 2, pp. 111-123.
- Chieng, W.H. and Hoeltzel, D.A. (1988.a): "A Combinatorial Approach to the Automatic Sketching of Planar Kinematic Chains and Epicyclic Gear Trains" in *Trends and Developments in Mechanisms, Machines and Robotics*, ASME DE-Vol. 15-1, pp. 79-90.
- Chieng, W.H. and Hoeltzel, D.A. (1988.b): "Computer-aided Kinematic Analysis of Planar Mechanisms Based on Symbolic Pattern Matching of Independent Kinematic Loops" in *Trends and Developments in Mechanisms, Machines and Robotics*, ASME DE-Vol. 15-1, pp. 79-90.
- Chieng, W.H. and Hoeltzel, D.A. (1989): "Systems for Unified Life-Cycle Mechanical Engineering Design: Shared-Tool Architectures vs. Distributed -Tool Architectures", submitted to 1989 NSF Engineering Design Research Conference, University of Massachusetts.
- Cox, B. J. (1986a): *Object Oriented Programming - An Evolutionary Approach*, Addison-Wesley Pub. Co., Reading, Massachusetts.
- Cox, B. J., (1986b): *Objects, Icons, and Software-IC's*, Byte Magazine, Aug.
- Dixon, J.R., Howe, A. Cohen, P.R. and Simmons, M.K. (1986): "DOMINIC:

- Progress Toward Independence in Design By Iterative Redesign" in *Proc. ASME Int. Computer in Engineering Conf*, Chicago, pp. 199-206.
- Dhande, S.G. and Chakraborty, J. (1978): "Mechanical Error Analysis of Spatial Linkages" in *ASME J. of Mechanisms, Trans. and Automation in Design*, Vol. 100, pp. 732-738.
- Dobrzanskyj, L (1966): *Doctoral Dissertation*, Columbia University.
- Fang, E. and Freudenstein, F (1988): "The Stratified Representation of Mechanisms" in *Trends and Developments In Mechanisms, Machines and Robotics*, ASME DE-Vol. 15-1, pp. 115-124.
- Fisher, I. and Freudenstein, F. (1984): "Internal Force and Moment Transmission in a Cardan Joint with Manufacturing Tolerances" in *ASME J. of Mech. Trans. and Auto. in Design*, Vol. ?, No. ? pp. 301-311.
- Freudenstein, F. (1959): "Structural Error Analysis in Planar Kinematic Synthesis" in *ASME J of Engineering for Industry*, Vol. 81, pp. 15-22.
- Freudenstein, F. and Maki (1986): "Development of An Optimum Variable-Stroke Internal Combustion Engine Mechanism From the Viewpoint of Kinematic Structure", *ASME J. of Mechanism*, Vol. 108, pp. 392-398.
- Gary, M.R. and Johnson, D.S. (1979): *Computer and Interactability - A Guid to the Theory of NP-Completeness*, W. H. Freeman and Co., New York.
- Groover, M.P. (1987): *Automation, Production, Systems, and Computer-Integrated Manufacturing*, Prentice-Hall Inc, New Jersey.
- Hinton, G.E. (1987): *Connectionist Learning Procedures*, Carnegie-Mellon technical report CMU-CS-87-115
- Hoeltzel, D.A., Chieng, W.H. and Zissmides, J. (1987): "Knowledge Representation and Planning Control in An Expert System for the Creative Design of Mechanisms" in *AI EDAM*, Vol. 1, No. 2, pp. 119-137.
- Hoeltzel, D.A. and Chieng, W.H. (1988.a): "An Adaptive, Generic Planning Model for Large Scale Integrated Engineering Design" in *Intelligent CAD System I*, Chapter 5, P.J.W. ten Hagen and Tomiyama, T. eds., Springer-Verlag, Netherland, pp. 72-89.
- Hoeltzel, D.A. and Chieng, W.H. (1988.b): "Statistical Machine Learning for Cognitive Selection of Nonlinear Programming Algorithms in Engineering Design Optimization" in *Trans. of Fifth Army Conf. on Applied Math. and Computing*, ARO report 88-1, pp. 65-88.
- Hoeltzel, D.A. and Chieng, W.H. (1988.c): "A Knowledge Based Approach to the Automated Mechanism Design Using Neural Network Computing" in *AAAI-88 Workshop on AI in Design* (book in press).
- Hoeltzel, D.A. and Cheing, W.H. (1989): "A unified Approach to the Kinematic Analysis of Joint Clearances and Link Length Tolerances for

- Determination of the Rotational and Positional Accuracy of Planar and Spatial Mechanism, To Appear.
- Hopfield, J.J. and Tank, D.W. (1985): "Neural Computing of Decisions in Optimization Problems, *Biological Cybernetics*, Vol. 52, pp. 141-152.
- Hu, M.K. (1962): "Visual Pattern Recognition by Moment Invariants" in *IRE Trans. Information Theory*, Vol. IT-8, pp. 179-187.
- Kitzmler, C.T. and Kowalik, J.S. (1987): "Coupling Symbolic and Numeric Computing in Knowledge-Based Systems" in *AI Magazine*, Issue of Summer 1987.
- Knowledge Craft Expert System Development Environment* (1986), The Carnegie Group.
- Kramer, S.N. and Sandor, G.N. (1975): "Slective Precision Synthesis - A General Method of Optimization for Planar Mechanism" in *ASME J of Engineering for Industry*, Vol. 97, pp. 689 - 701.
- Lawler E.L., Lenstra, J.K. and Kan, R. (1982): "Recent Developments in Deterministic Sequencing and Scheduling: A Survey" in *Deterministic and Stochastic Scheduling*, Dempster, M.A.H. ed., D. Reidel Pub. Co., North Holland.
- Lippmann, R.P. (1987): "An Introduction to Computing with Neural Nets" in *IEEE ASSP Magazine*, issue of April 1987
- McDermott, J. (1982): "R1: A Rule-Based Configurer of Computer Systems", in *Artificial Intelligence*, Vol. 19, pp. 39-88
- Morgan, A.P. (1987): *Solving Polynomial Systems Using Continuation for Engineering and Scientific Problems*, Prentice-Hall Publishers.
- Nikravesh, P.E. (1988): *Computer-Aided Analysis of Mechanical Systems*, Prentice Hall, New Jersey.
- Olson, D.G., Thompson, T.R., Riley, D.R. and Erdman, A.G. (1984): "An Algorithm for Automatic Sketching of Planar Kinematic Chains" in *ASME J. of Mechanisms*, 84-DET-108.
- Papalambros, P.Y. and Wilde, D. J. (1988): *Principle of Optimal Design - Modeling and Computing*, Cambridge University Press, New York.
- Preparata, F.P. and Shamos, M.I. (1985): *Computational Geometry - An Introduction*, Springer-Verlag, New York, pp. 179-256.
- Paul, B. (1979): *Kinematics and Dynamics of Planar Machinery*, Prentice Hall, New Jersey.
- Schrijver, A. (1986): *Theory of Linear and Integer Programming*, John Wiley & Sons, New York.
- Shigley, J.E. (1977): *Mechanical Engineering Design*, 3rd Edition, McGraw-Hill Book Co., New York.
- Shortliffe, E.H. (1976): *Computer-Based Medical Consultations: MYCIN*, American Elsevier, New York.
- Sohn, W. and Freudenstein, F. (1986): "A Application of Dual Graphs to the

- Automatic Generation of Kinematic Structures of Mechanisms" in *ASME J. of Mechanisms, Trans. and Automation in Design*, Vol. 108, pp. 392-402.
- Ullman, D.G. and Dietterich, T.A. (1986): "Mechanical Design Methodology: Implementation on Future Developments of Computer-aided Design and Knowledge-based Systems" in *Proc. of ASME Int. Computers in Engineering Conf.* Chicago, pp. 173-180.
- Vanderplaats, G.N. (1984): *Numerical Optimization Techniques For Engineering Design*, McGraw-Hill Book Co., New York.
- Vollbracht, G.T. (1986): "The Time for CAEDM is Now" Structural Dynamics Research Co., Technical Report, referenced in *Computer-Aided Engineering*, CAD/CAM section, Vol. 7, Oct., 1988, pp. 28.
- Woo, L.S. (1969): "An Algorithm for Straight-Line Representation of Simple Planar Graphs" in *J. of Franklin Institute*, pp. 197-208.
- Yan, H.S. and Huang W.M. (1983): "A Method for Identification of Planar Linkage Chains", *ASME J. of Mechanisms, Trans. and Auto. in Design* Vol. 105, No. 4, pp. 658 -662.

A HIERARCHICAL METHODOLOGY
FOR THE
CREATIVE DESIGN OF MECHANISMS

Wenchun Eugene Fang

(with the guidance of Professor F. Freudenstein)

Submitted in partial fulfillment of the
requirement for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

1989

ABSTRACT

A Hierarchical Methodology for the Creative Design of Mechanisms

Wenchun Eugene Fang

A new methodology for the creative design of mechanisms has been developed. A hierarchy is established which is used both to represent the kinematic structure of each mechanism and for the classification of different mechanisms. The consistency between the representation and the classification provides a new perspective to many problems during the creative design of mechanisms and has proven a powerful tool in manipulating a whole class of mechanisms at once. Using this new concept, the following goals are achieved:

- 1. The kinematic structure of a mechanism can be "spelled-out" by a series of characters, called the stratified code. This code can be used for automatic identification as well as a key for a dictionary-like arrangement of collections of mechanisms.*
- 2. A systematic and fully-automated procedure for exhaustive enumeration of mechanisms was developed. The efficiency of this new procedure is orders-of-magnitude higher than that of traditional procedures.*
- 3. A fully-automated procedure for mechanisms sketching from the stratified code was developed. Sketches generated this way are free of crossing links, properly-dimensioned, and have acceptable transmission angles.*

Using this hierarchical approach, millions of mechanisms have been generated. A dictionary of mechanisms sorted by their stratified codes is included in the Appendix. The hierarchical approach has provided elegant solutions to mechanisms identification, enumeration and sketching. Other mechanism design problems should benefit as well if this new approach is adopted.

© 1989

Wenchun Eugene Fang

ALL RIGHTS RESERVED

CONTENTS

CONTENTS	i
FIGURES	iii
TABLES	iv
ALGORITHMS	v
ACKNOWLEDGMENTS	vi
CHAPTER 1 INTRODUCTION	1
CHAPTER 2 PREVIOUS WORK	4
CHAPTER 3 BASIC CONCEPTS AND TERMINOLOGIES	9
3.1 GRAPH THEORY AND THE KINEMATIC STRUCTURE OF MECHANISMS	9
3.2 THE OPTIMUM-CODE APPROACH TO MECHANISM IDENTIFICATION	12
3.3 THE CONTRACTION MAPPING OF A GRAPH	14
3.4 LINKED LISTS AND ARRAYS	15
3.5 NOTATION	16
CHAPTER 4 REPRESENTATION OF MECHANISMS	18
4.1 INTRODUCTION	18
4.2 LEVELS OF ABSTRACTION IN REPRESENTATION	19
4.3 THE STRATIFIED ADJACENCY MATRIX	21
4.4 COMPARISON BETWEEN LINKED LISTS	26
4.5 CLASSIFICATION BY CONDENSATION	28
CHAPTER 5 IDENTIFICATION OF MECHANISMS	31
5.1 INTRODUCTION	31
5.2 REDUCTION OF VERTEX SEQUENCE	33
5.3 SCHEME FOR OBTAINING THE STRATIFIED CODE OF A GRAPH	38
5.4 SUMMARY OF PROCEDURES FOR THE IDENTIFICATION OF KINEMATIC STRUCTURES	39
5.5 EXAMPLES	41
CHAPTER 6 ENUMERATION OF MECHANISMS	45
6.1 INTRODUCTION	45

6.2	ENUMERATION BY EXPANSION	47
6.3	ENUMERATION OF CONTRACTED GRAPHS	50
6.3.1	Primary Contracted Graphs	52
6.3.2	Secondary Contracted Graphs	53
6.4	ENUMERATION OF MONOCHROME GRAPHS	54
6.5	ENUMERATION OF COLORED GRAPHS	60
6.6	SUMMARY OF ENUMERATING PROCEDURES	62
6.7	EXAMPLES	65
CHAPTER 7	SKETCHING OF MECHANISMS	70
7.1	INTRODUCTION: A HIERARCHICAL APPROACH	70
7.2	CONTRACTED LEVEL: CROSSING-FREE SKETCH	73
7.2.1	Properties of the Stratified Adjacency Matrix	73
7.2.2	Determination of the Peripheral Loop	74
7.3	MONOCHROME LEVEL: ACCEPTABLE PROPORTIONS ...	77
7.3.1	Determination of Coordinates of Peripheral Vertices	78
7.3.2	Determination of Coordinates of Internal Vertices	80
7.3.3	Determination of String Shapes	81
7.4	COLORED LEVEL: FAVORABLE ORIENTATIONS	83
7.4.1	Patterns for Sketching Revolute Pairs	84
7.4.2	Patterns for Sketching Mixed Pairs	88
7.5	SUMMARY OF SKETCHING PROCEDURE	89
7.6	CONCLUSION	91
CHAPTER 8	CONCLUSION	93
8.1	LIMITATIONS AND EXTENSIONS	93
8.1.1	Single-Loop Mechanisms	93
8.1.2	Joint Types Other than R and P	96
8.1.3	Specifying the Frame, Drive, and Other Links	96
8.1.4	Specifying the Dimensions of Mechanisms	99
8.2	SYNOPSIS	99
REFERENCES	102
APPENDIX	105

FIGURES

Figure 1. Graph and Corresponding Kinematic Structure	10
Figure 2. Relabelling the Graph of the Stephenson Chain	13
Figure 3. Contraction Mapping	14
Figure 4. Levels of Abstraction	21
Figure 5. Traversal of a Nested List for Comparison	27
Figure 6. Classification by Condensation	29
Figure 7. Permutation of Vertices	35
Figure 8. Isomorphism of Aircraft Landing Gear Mechanisms	41
Figure 9. Primary Contracted Graphs	50
Figure 10. Enumeration of Contracted Graphs	51
Figure 11. Enumeration of Monochrome Graphs from a Contracted Graph	65
Figure 12. Determination of the Peripheral Loop	74
Figure 13. String Ratio and its Effects	79
Figure 14. Arc Offset	82
Figure 15. Patterns for Sketching Strings of R and P Joints	84
Figure 16. Effects of Regularity	85
Figure 17. Effects of Slenderness	87

TABLES

Table I. Elements of the Stratified Adjacency Matrix	24
Table II. Expansion of Elements	55
Table III. Expansion of Strings	60
Table IV. Summary of Enumeration Results	62
Table V. Summary of Enumeration Results (Continued)	62
Table VI. Expansion of Strings with Geared Joints	96
Table VII. Comparison Between the New and Traditional Approaches . . .	100

ALGORITHMS

Algorithm 1. Best-Only Permutation of Vertices	34
Algorithm 2. Further Trimming of Vertex Permutations	37
Algorithm 3. Coding Algorithms	39
Algorithm 4. Distribution of Binary Vertices	57
Algorithm 5. Expansion to Monochrome Level	58

ACKNOWLEDGMENTS

I am heavily indebted to Prof. F. Freudenstein, my advisor. This dissertation would never have been finished without his technical guidance, as well as his support and encouragement. I am fortunate to have such an advisor during the last four years.

Thanks are due to Prof. T. R. Chase at University of Minnesota, who opened my eyes to kinematics five years ago. His patience and guidance are greatly appreciated.

I would like to thank Prof. E. Tuttle at University of Denver, whose comments on my 1988 ASME Mechanisms Conference paper helped me to identify an error in the enumeration process. Without her input, this error might have been included in this dissertation.

Finally and most importantly, I would like to thank my wife, Ya-Ping. If I have achieved anything, it is because she takes care of everything and endures my negligence. My deepest love and appreciation are to her.

CHAPTER 1 INTRODUCTION

The process of mechanism design can be subdivided into two stages: synthesis and analysis. The synthesis process can be subdivided into two additional stages: type synthesis and dimensional synthesis. The former concerns the process of creating the mechanism (e.g. four-bar linkages, slider-crank mechanisms, etc.), and the latter concerns the process of specifying mechanism dimensions (e.g. the length of the links, etc.). While most analytical work pertains to the dimensional-synthesis stage, the process of type synthesis - also called creative design or conceptual design - remains mostly unexplored, i.e., relegated to the intuition and experience of the designer.

Traditionally, engineers explore atlases of mechanisms during the creative design stage. However, since collections of atlases are far from exhaustive, and mechanisms are classified according to function, the design process is both time-consuming and the result highly dependent on the experience as well as the inspiration of the designer. Moreover, the informal, intuitive classification schemes used in these atlases make automatic information storage and retrieval difficult, let alone the automation of the design process itself. Since creative design is one of the earliest stages of the total design process, it constitutes the foundation of all other

design efforts. If a poor choice is made during the creative-design stage, it can never be recovered in the course of subsequent synthesis, analysis, or optimization. Studies in creative design, therefore, are of strategic importance amongst all design methodologies.

A rational approach to creative design based on graph theory has been investigated for decades. However, existing solutions to some fundamental questions such as how to identify, enumerate, and sketch mechanisms automatically have been less than satisfactory. Most difficulties in this area can be attributed to the lack of a consistent and reliable representation of the kinematic structure of mechanisms. The investigation, therefore, is concerned with the development of a representational formalism of kinematic structures suitable for solving the above-mentioned problems.

A new scheme for the representation of kinematic structures of mechanisms is developed (Chapter 4). It basically replaces the traditional adjacency matrix of a graph by a new "stratified adjacency matrix", which has smaller rank but more complex elements. This scheme implies a rational method of mechanism classification, gives a new perspective to problems of creative design of mechanisms, and hence renders elegant and consistent solutions. Three sub-problems in the creative design of mechanisms are explicitly solved in this thesis:

1. The identification or isomorphism problem (Chapter 5) is solved by permuting the rows and columns of the stratified adjacency matrix to a

canonical form, called the "stratified code", and isomorphism is determined by comparing the stratified code. In contrast to the present state-of-the-art, which can be either efficient or reliable, but not both, the new solution to the identification problem is both efficient and reliable.

2. The enumeration problem (Chapter 6) is solved by incrementally refining the representation of each "class" of mechanisms until enough details of kinematic structures are specified. The isomorphism checking procedure is substituted by a more efficient canonical-form validation procedure. Mechanisms generated this way are already sorted in sequential order. As a result, the new enumeration method is a few orders-of-magnitude more efficient than traditional ones.
3. The sketching problem (Chapter 7) is solved by determination of the common geometric properties of a "class" of mechanisms from their common kinematic structure, and successively refining the geometries according to the refined structures. This collective approach minimizes the required computation during the sketching of a large number of mechanisms, and the resulting geometric similarities within a class of mechanisms are very helpful for the designer's visualization of the structural similarities.

CHAPTER 2 PREVIOUS WORK

Beginning in the mid-sixties, the creative design of mechanisms has been investigated with the aid of graph theory. The first rational study of type synthesis using graph theory was described by Freudenstein and Dobrjanskyj [10] in 1964. In this investigation, it was proposed that type synthesis be accomplished systematically by means of the following steps: determination of the degree of freedom and number of desired links; enumeration of the corresponding kinematic chains; and enumeration of the mechanisms satisfying the given kinematic specifications. Graphs of similar local degree structure are partitioned into "groups" and enumerated separately. A number of matrices of incidence of graphs were also defined and their relationships developed. In this study 16 single-degree-of-freedom, eight-link kinematic chains were enumerated.

In 1964, Crossley [6] developed an algorithm to enumerate two degree-of-freedom, nine-link kinematic chains. Later [7], Crossley also used graph theory in the enumeration of mechanisms of eight links or less and determined the number of single degree-of-freedom kinematic chains of four, six, and eight links to be 1, 2, and 16, respectively.

In 1967, Dobrjanskyj and Freudenstein [8] developed a heuristic algorithm for the isomorphism testing of kinematic chains, a method for the automatic sketching of graphs of mechanisms and systematically enumerated single-loop, spatial mechanisms. These developments showed the value of a rational approach to type synthesis: the potential for automation.

In 1967, Woo [32] introduced the concept of contraction mapping of graphs, which basically combines a string of edges and binary vertices into a single edge. Enumeration was performed in two stages: the contraction mappings are enumerated first; then for each contraction mapping, binary vertices are superimposed so that the total number of vertices and edges satisfy the degree-of-freedom equation. Using this method, Woo enumerated 230 single-degree-of-freedom, ten-link kinematic chains.

In 1970, Buchsbaum and Freudenstein [4] proved that "the sub-graph of a geared kinematic chain formed by deleting all edges representing gear pairs, form a tree" based on the requirement of constant center distance of meshing gears and unlimited rotatability. The concept of transfer vertices was also introduced. These properties simplify the enumeration process substantially. As a result, geared kinematic chains are the best studied among all kinematic chains with two or more types of joints. Gear trains with up to six links have been enumerated in [4,29]. Recently, the technology has been extended to the design of automatic transmissions [30].

In 1975, Uicker and Raicu [31] conjectured that the characteristic polynomial of the adjacency matrix of a graph is unique. Although this conjecture is not applicable to general graphs, it does apply to graphs of bar-linkages with up to eight links. When it failed to identify more complex mechanisms [1,2,25], various modifications were proposed. In 1982 Yan and Hwang [35] proposed that the characteristic polynomial of the linkage structure matrix is unique up to isomorphism. In 1987 Mruthyunjaya and Balasubramanian [19] suggested that the characteristic polynomial of the degree matrix is more discriminatory than the characteristic polynomial of the adjacency matrix. In spite of these modifications, the characteristic-polynomial approach has not been a sufficiently reliable identification method for kinematic structures, because it lacks a sound theoretical foundation.

In 1974, Shah et. al. [24] introduced the optimum-code of a graph which was virtually overlooked by kinematicians until similar concepts called Max-code [1] and Min-code [2] were introduced more than ten years later. Basically, it involves the permutation of the vertex labeling of the graph so that the binary sequence (the code) of the upper triangular adjacency matrix becomes a maximum or minimum. Noting that the number of possible permutations is proportional to the factorial of the number of vertices of a graph, heuristics is necessary in order to improve the efficiency. Heuristic algorithms, however, may lead to local maxima or minima [2] and destroy the robustness of the entire identification scheme. Hence an implementation of this method would face a "robust versus efficiency" dilemma.

However, there are some virtues of this approach: The kinematic structure is retrievable from the code, and a collection of codes of kinematic chains can be sorted in a sequential manner. Retrievability is desirable because it frees the computer from the need to store the representation of the mechanism (usually the adjacency matrix) and the identifier (e. g. the characteristic polynomial) simultaneously. Sortability makes a dictionary-like collection of mechanisms possible. For these reasons, the optimum-code approach is generally considered favorable.

In 1979, Freudenstein and Maki [12] proposed and outlined the creative design procedure according to the concept of the separation of structure and function. The power of this approach was further demonstrated in the process of designing mechanisms for variable-stroke engines [9].

In 1984, Olson et. al. [20] developed an automatic sketching algorithm for bar linkages which applied a scheme of concentric rings to avoid crossing links. User interaction, however, is needed to guarantee the non-crossing feature.

In 1986, Sohn and Freudenstein [25] applied the concept of dual graphs and proposed a new algorithm for the enumeration of kinematic structures. The dual graphs were constructed first, the edges of the dual graphs labelled next and finally conventional graphs were enumerated from the labelled dual graphs. This procedure is extremely efficient although it is limited to planar graphs only.

In 1988, Chieng and Hoeltzel [5] enumerated all possible "in", "out", or "on" relationships between loops of a graph. An exhaustive tree-search algorithm was developed to ensure non-crossing links for the automatic sketching of mechanisms with revolute and prismatic pairs. An algorithm for gear train sketching was developed as well.

CHAPTER 4 REPRESENTATION OF MECHANISMS

4.1 INTRODUCTION

A mechanism cannot be studied until it is sketched, described by an accompanying text, or "represented" in a way which is understandable for the reader. Similarly, there can be no computer-aided-design of mechanisms unless the mechanisms can be represented by integers, arrays, linked lists, or other appropriate data structures. The representational scheme is important, because it is the foundation of all operations during the design process. Different data structures usually imply different algorithms for a design task, which in turn determine the efficiency of the entire design process.

In this chapter, a new representation of the kinematic structures of mechanisms is introduced. The underlying philosophy of the new approach is to keep intact the fundamental data structure -- which can be arrays or linked lists -- and to specify only the minimum amount of kinematic structure at all times. The latter is achieved by means of a concept called "levels of abstraction" described in Sec. 4.2, while the former is accomplished by representing a graph by the "stratified adjacency matrix" described in Sec. 4.3. Elements of the stratified adjacency

matrices are imbedded in linked lists, which can be sorted according to the rules defined in Sec. 4.4. Implications of this representational scheme for the classification of mechanisms are delineated in Sec. 4.5.

4.2 LEVELS OF ABSTRACTION IN REPRESENTATION

During a discussion of the kinematics of a mechanism, many properties essential to its construction are usually tacitly ignored. Such properties may include the material of which the mechanism is made, actual link geometry, etc. Similarly, during the process of creative design, actual link dimensions are omitted. When graph theory is employed in the study of kinematic structures, the locations of the frame and drive links are usually ignored as well. The above simplifications are made in order to separate the most relevant characteristics of the mechanisms from others and hence to clarify the problem of interest.

Along the same lines of reasoning, it may be observed that not all details of the kinematic structure are needed during each step of the creative design process. Neglecting some unnecessary structural properties may bring the immediate objective to a clear focus. If the minimum number of structural properties are used at each step of creative design, the resulting process should be more efficient.

Different requirements for the degree of detail of kinematic structures will result in different, though closely related, graphs. These different graphs of the same mechanism are at different levels of abstraction. Graphs at a concrete level will preserve more kinematic structures, while graphs at an abstract level preserve less. The process of obtaining a more abstract graph from a given graph is called condensation, while the process of obtaining graphs which are more concrete than the given graph is called expansion. Four levels of abstraction are defined here for later discussions (from concrete to abstract):

Colored: The colored graph is the traditional graph representation of mechanisms. At this level, each link of the mechanism is designated by a vertex; and each joint designated by an edge. The type of joint is designated by the label of each edge.

Monochrome: The monochrome graph is similar to the colored graph except that there are no edge labels, meaning that all types of joints of a mechanism are represented in the same manner.

Contracted: The contracted graph is the contraction mapping of the monochrome graph (see Sec. 3.3). In other words, only links having three or more joints are designated by vertices, and the relationship between these major links are designated by the number of strings (see Sec. 3.1) of joints and binary links.

Simplified: The simplified graph is similar to the contracted graph, except that multiple strings between two major vertices are treated as one string.

For example, graphs of the variable-stroke engine mechanism (see Figure 1, p. 10) at the four levels of abstraction are shown in Figure 4.

4.3 THE STRATIFIED ADJACENCY MATRIX

A straightforward method of representing graphs at various levels of abstraction is the traditional adjacency matrices. For example, the adjacency matrices of the graphs of Figure 4 (a) and (b) are 8 x 8 square matrices given by eqs. (3-1) and (3-2), while the adjacency matrices of the graphs of Figure 4 (c) and (d) may read (see Sec. 3.3):

$$AM(c) = \begin{matrix} & \begin{matrix} A & B & C \end{matrix} \\ \begin{matrix} A \\ B \\ C \end{matrix} & \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix} \end{matrix} \quad (4-1c)$$

$$AM(d) = \begin{matrix} & \begin{matrix} A & B & C \end{matrix} \\ \begin{matrix} A \\ B \\ C \end{matrix} & \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \end{matrix} \quad (4-1d)$$

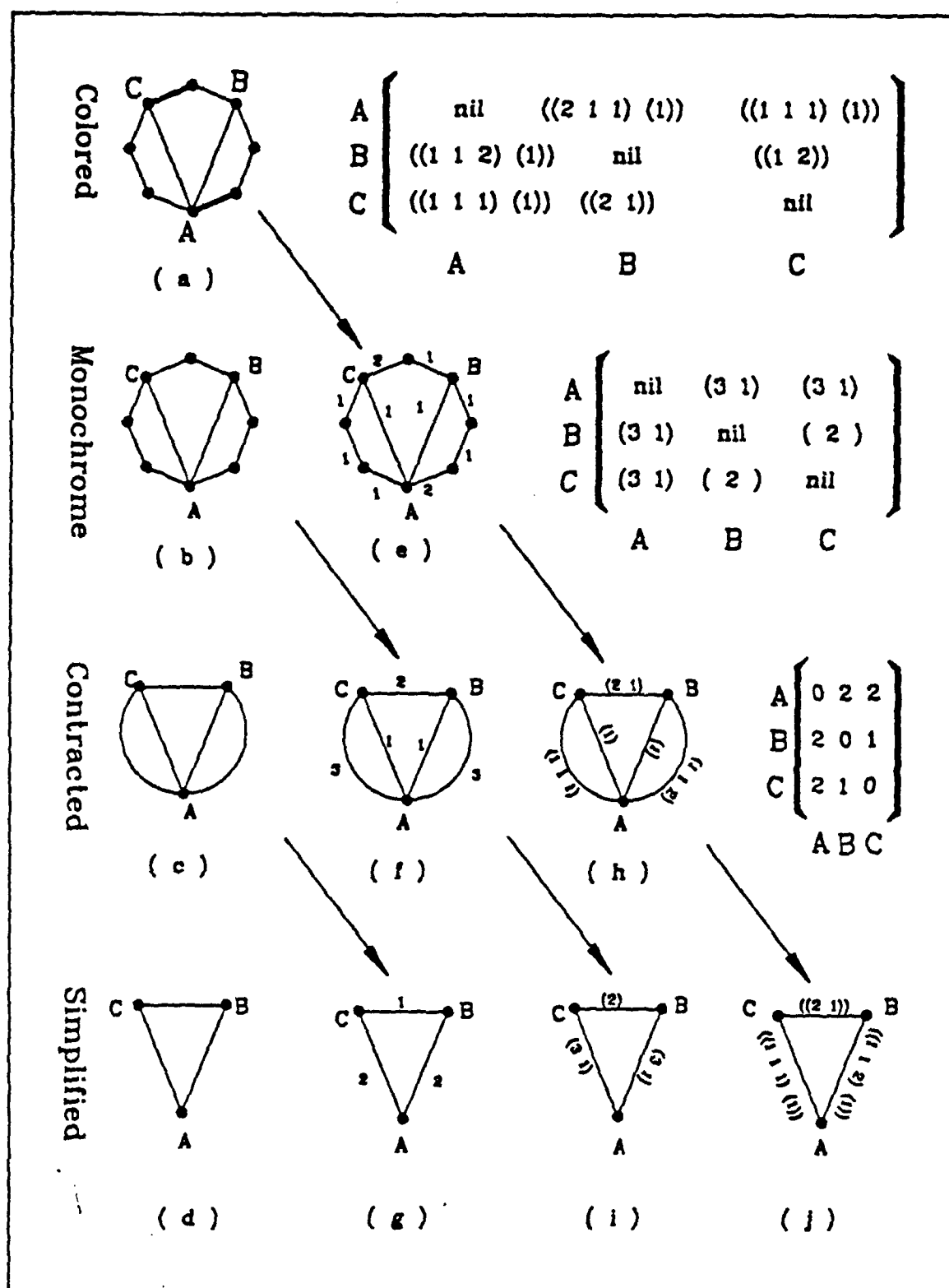


Figure 4. Levels of Abstraction

This intuitive approach, however, requires the conversion of matrices of different ranks during condensation or expansion between monochrome and contracted levels. Since condensation and expansion are constantly required during creative design, the resulting conversions are both confusing and inefficient. A new representational scheme, which ensures the rank of the adjacency matrices remain constant for all levels of abstraction, is devised to expedite this process.

Consider the graphs in Figure 4 (c) and (d). If an integer label is attached to the edges of the simplified graph to designate the multiplicity (see page 14) between the two incident major vertices, the graph of Figure 4 (g) will preserve exactly the same amount of structural information as does Figure 4 (c). Similarly, if the edges of Figure 4 (c) are labelled according to the length of the corresponding strings of Figure 4 (b), the resulting Figure 4 (f) can be treated as identical to Figure 4 (b). Moreover, This edge labelling technique can be carried out recursively so that a graph with a simple structure and complex labels, such as Figure 4 (i), can contain exactly the same amount of information as the graph of a complex structure without labels, such as Figure 4 (b).

In the spirit of this example, definitions of elements of the stratified adjacency matrices of monochrome and colored graphs are summarized in Table I. (The stratified adjacency matrices for simplified and contracted graphs are identical to traditional adjacency matrices.) Basically, the adjacency matrix of the contracted graph is kept intact, more detailed structural information is stored by delineating

more details of the strings and edges inside each matrix element. Linked lists, which are highly flexible and easily adapted to computer implementations, are therefore used to represent these elements.

The "major diagonal" of Table I defines the meaning of integers used to designate a certain part of a graph (e.g. edges, strings, see p. 10). The more abstract the graph is, the larger part of the graph an integer can represent. The representation of an element a_{ij} of a stratified adjacency matrix is defined recursively as a list of strings connecting vertex i and j , and a string is recursively represented by a list of its constituting edges.

It should be noted that when strings are listed together in an element of the stratified adjacency matrix, the order or sequence is immaterial as long as representation of the kinematic structures is concerned. In contrast, when a string is represented by a list of edges, the sequence of edges is very important because different edge sequences will in general represent different kinematic structures. It is also worth noting that the edge sequence will be reversed depending on whether this involves traversing from vertex i to j or the reverse. For example, the three-edged string from vertex A to B in Figure 4 (a) should be the list (2 1 1), while the same string should be represented by the list (1 1 2), if traversing from vertex B to A. By way of comparison, in Figure 4 (b), either the list (3 1) or (1 3) might be used to represent the two strings connecting vertices A and B regardless of which direction is used for the traverse. This ambiguity is undesirable for the purpose of

Table I. Elements of the Stratified Adjacency Matrix

	element a_{ij}	string = a component of an element	edge = a component of a string
contracted level	integer = multiplicity = number of strings between vertices i and j	ignored	ignored
monochrome level	list of strings*	integer = length of string = number of edges in the string	ignored
colored level	list of strings* = nested list of edges	list of edges (in the order of traverse from vertex i to vertex j)	integer = type of joint (1=revolute 2=prismatic 0=none)

*sorted in non-increasing order; see Sec. 4.4 for definition of comparison of linked lists.

graph identification. Rules for the comparison of linked lists are defined in the next section such that strings can always be sorted in descending order in an element. Thus each element of the stratified adjacency matrix is uniquely defined.

When the major vertices are permuted, the location of each element of the stratified adjacency matrix will in general change, but each internal element will remain intact. While different levels of abstraction may change the complexity of each element, the basic structure of the matrix need not be perturbed.

It may be noted also that the stratified adjacency matrix is symmetric about its major diagonal at the monochrome and more abstract levels. At the colored level, it may be considered "skew" symmetric with the understanding that "skew" means the reversal of the strings of the corresponding elements. Either way, it is always sufficient to represent a graph by the upper or lower triangular part of its stratified adjacency matrix.

4.4 COMPARISON BETWEEN LINKED LISTS

For the purpose of determining a sequence, any definition of list comparison is adequate as long as it is self-consistent (e.g. if $A > B$ and $B > C$ then $A > C$). Two rules may seem natural and are already widely used: the rules used in dictionaries to sort words, and the rules used to compare two decimal numbers.

5.3 SCHEME FOR OBTAINING THE STRATIFIED CODE OF A GRAPH

When the Best-Only permutation is carried out, it can be observed that the sequence of the largest labels of connecting edges follows a path from upper left to lower right just one element off the major diagonal of the stratified adjacency matrix. If the main diagonal is called the zeroth diagonal and these $N-1$ elements the first diagonal, the D-th diagonal may be defined in terms of the element $a_{d,0}$ at the upper left proceeding diagonally down to the element $a_{(n-1),(n-d-1)}$ at the lower right, the difference of column number and row number of each element being always equal to D . A stratified adjacency matrix is then expressible by concatenating the second diagonal behind the first, the third diagonal behind the second, etc. The resulting sequence is a "code" of the matrix embedded in a nested list, which can be compared by the list comparison rules given previously.

For each possible permutation generated by the Best-Only Algorithm (see p. 35), there is a corresponding code. In view of our objective -- a canonical form of the adjacency matrix independent of the initial choice of the sequence of rows and columns -- we can arbitrarily define that only those sequences that will form the largest code are valid. Other sequences are simply discarded. This process is delineated in Algorithm 2. The largest code of edge labels of the stratified adjacency matrix obtained this way is called the stratified code of a graph. Vertex

permutations which produce the stratified code are called valid vertex sequences. One of the valid vertex sequences is chosen as the default vertex sequence. Every other valid vertex is called alternative vertex sequence and is expressed as a permutation of the default sequence. The converting process between the stratified matrix and the stratified code is straightforward and is described in Algorithm 3.

It is evident that if two graphs have the same stratified code then they are isomorphic, because they will have exactly the same adjacency matrix. The reverse statement (if two graphs are isomorphic then they have the same stratified code), is also true because all the N factorial vertex sequences have been considered in the Best-Only Algorithm. The only difference between a brute-force generation of all N factorial codes for a given matrix and the proposed strategy is that the latter drops off most of the N factorial codes at the earliest possible stage. Hence the identity of the stratified code is a necessary and sufficient condition for the graph isomorphism.

5.4 SUMMARY OF PROCEDURES FOR THE IDENTIFICATION OF KINEMATIC STRUCTURES

Given a set of graphs of mechanisms, the new process of isomorphism testing is summarized as follow:

1. Express each graph in the form of a stratified adjacency matrix.

function encode(SAM, P)

This function returns the stratified code of a graph given by its stratified adjacency matrix, SAM, P is any valid vertex sequence.

```

CODE := empty
for D := N-1 to 1 do begin
  for j := N-D-1 to 0 do begin
    I := J+D
    push SAM( P(I), P(J) ) into CODE
  end
end
return CODE

```

function decode(CODE)

This function transform the stratified code into a stratified adjacency matrix.

```

Initialize an N x N empty array SAM
for D := 1 to N-1 do begin
  for J := 0 to N-D-1 do begin
    I := J+D
    SAM(I,J) := first( CODE )
    SAM(J,I) := sort( reverse( first(CODE) ) )
    CODE := rest( CODE )
  end
end
return SAM

```

Algorithm 3. Coding Algorithms

2. For each stratified adjacency matrix, obtain the stratified code.
 - 2.1 Apply Algorithm 1 to get an initial set of possible sequences.
 - 2.2 Apply Algorithm 2 to reduce the number of valid vertex sequences.
 - 2.3 Any remaining sequences of step 2.2 can be fed into algorithm 3 for the stratified code.
3. Check the identity of the resulting stratified codes, two graphs are isomorphic if and only if the stratified codes are identical.

5.5 EXAMPLES

The well known Artobolevsky's atlas [3] Sec. IV, part 21, (pp. 416-444), shows thirty aircraft landing gear mechanisms. Two of these are eight-link mechanisms (No. 1425 and 1435, see Figure 8). Do any of these have the same kinematic structure? This question can be answered by following the step-by-step guideline given in the previous section (The frame of the aircraft is labelled as link 0 in all mechanisms).

1. Express the graph of each mechanism in the form of a stratified adjacency matrix.

There are three links of degree greater than two in each mechanism, namely, links 2, 3, 7 for mechanism (A) and links 0, 3, 5 for mechanism (B). In case (A), links 2 and 3 are connected directly by a revolute joint and a series of

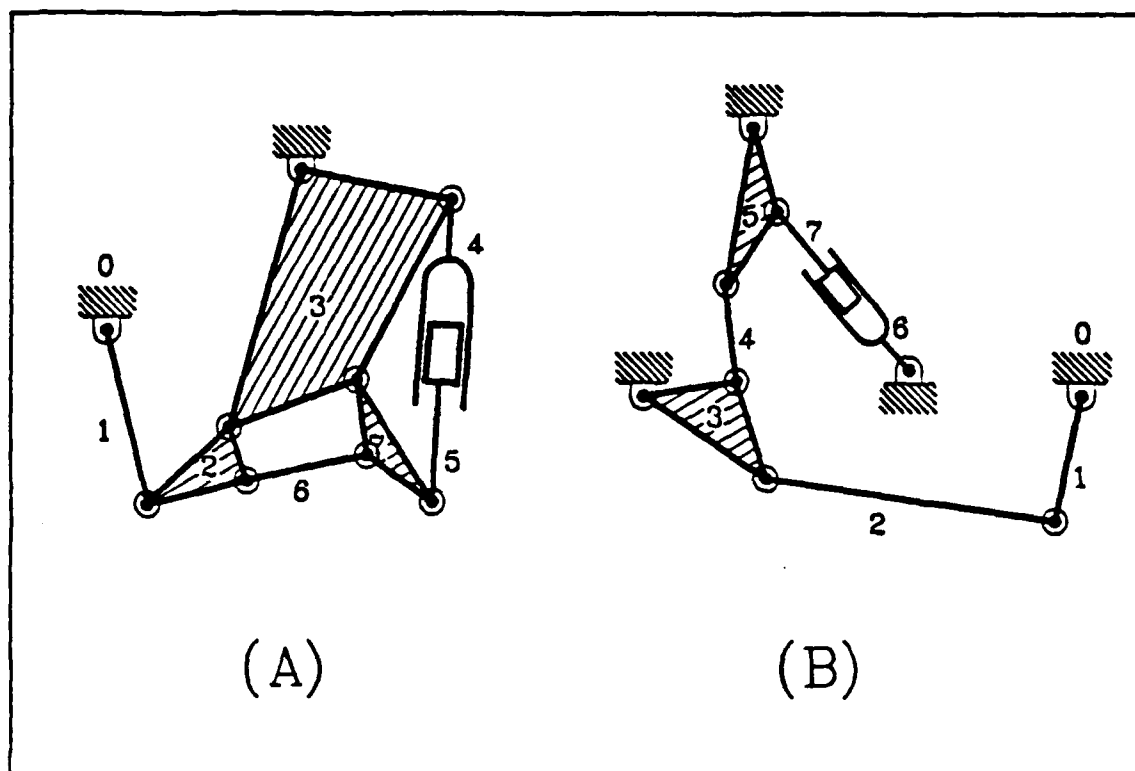


Figure 8. Isomorphism of Aircraft Landing Gear Mechanisms

binary links and revolute joints (links 0 and 1). The former is represented by string (1), the latter by string (1 1 1). Hence, element 3-2 of the adjacency matrix should be $((1 \ 1 \ 1) \ (1))$. Similarly, there are two strings connecting links 3 and 7, as well as a prismatic joint between the two connecting binary links (link 4 and 5). Element 7-3 of the stratified adjacency matrix becomes $((1 \ 2 \ 1) \ (1))$. The stratified adjacency matrix for mechanism A may now be constructed:

$$A = \begin{matrix} & 2 & & & \\ & 3 & & & \\ & 7 & & & \\ & & & & \end{matrix} \begin{bmatrix} 0 & ((111)(1)) & ((11)) \\ ((111)(1)) & 0 & ((121)(1)) \\ ((11)) & ((121)(1)) & 0 \\ 2 & 3 & 7 \end{bmatrix}$$

Similarly, the stratified adjacency matrix for mechanism B becomes:

$$B = \begin{matrix} & 3 & & & \\ & 5 & & & \\ & 0 & & & \\ & & & & \end{matrix} \begin{bmatrix} 0 & ((11)) & ((111)(1)) \\ ((11)) & 0 & ((121)(1)) \\ ((111)(1)) & ((121)(1)) & 0 \\ 3 & 5 & 0 \end{bmatrix}$$

2. For each stratified adjacency matrix, obtain the stratified code.

2.1 Apply Algorithm 1 to obtain an initial set of possible sequences.

(A). The largest element $((121)(1))$ is between link 3 and link 7, Therefore two sequences starting with (3 7) or (7 3) should be investigated. The only way to extend sequence (7 3) is (7 3 2), with a new edge label $((111)(1))$. The only way to extend sequence (3 7) is (3 7 2), which produces a new edge label $((11))$. Since edge label $((111)(1))$ is greater than $((11))$, sequence (7 3 2) is the only possible sequence generated in this step.

(B). Similarly, Algorithm 1 generated just one sequence: (5 0 3).

2.2 Apply Algorithm 2 in order to reduce the number of sequences.

In this example, this step is trivial since only one sequence has been generated in the previous step.

2.3 Any remaining sequence of step 2.2 can be fed into algorithm 3 for the stratified code.

(A). Basically, the stratified code is simply the sequence of elements of the permuted stratified matrix, starting from the upper-left element of the first diagonal, which is ((121)(1)) in this example, proceeding down to the lower-right most element, which is ((111)(1)) in this example. The second diagonal is concatenated to the first diagonal in the same manner. There is only one element in the second diagonal in this example: ((11)). The resulting stratified code should read: (((121)(1)) ((111)(1)) ((11))).

(B). Similarly, stratified code = (((121)(1)) ((111)(1)) ((11)))

3. Check the identity of the resulting stratified codes, two graphs are isomorphic if and only if the stratified codes are identical.

Obviously, the two codes are identical, so the two graphs of the mechanisms are isomorphic.

CHAPTER 6 ENUMERATION OF MECHANISMS

6.1 INTRODUCTION

Techniques for the enumeration of all mechanisms up to a certain order of complexity have been studied for decades. Earlier researchers were preoccupied with how to create rationally an atlas of mechanisms for designers' reference. Nowadays expert systems for design automation are beginning to emerge, and these require even more powerful techniques for on-line enumeration of mechanisms. The highlight, however, has been shifted to techniques of selective enumeration: the ability to generate only mechanisms which will satisfy some given specifications.

Difficulty in enumeration is two-fold: both the number of mechanisms and the time to identify a mechanism grows exponentially with the complexity of mechanisms (indicated by, say, the number of links). The enumeration of mechanisms therefore requires a time proportional to the product of two exponential functions and consequently, can be devastating even for very powerful computers if algorithms or their implementations are not well-poised.

A graph has to satisfy a number of restrictions to represent a valid kinematic structure. These restrictions include: The graph must be bi-connected, the number of

vertices and joints must satisfy the general degree-of-freedom equation, and no subgraph can have zero or less degrees-of-freedom. Since only a tiny fraction of graphs would satisfy these restrictions, the naive approach of generating all theoretically possible graphs and filter out undesirable ones would be hopelessly slow. Consequently, a practical approach which would directly generate graphs satisfying at least some of the restrictions is indispensable.

Due to the unsurmountable time complexity involved, traditional approaches to the enumeration of kinematic structures have been static in nature: Graphs of a certain complexity are generated once and for all, constructing an atlas for later reference, thus by-passing this complex computation in all later calculations. Nevertheless, the amount of memory required to store the atlas grows out of control very fast when this approach is adopted, leaving it as unsolvable as before.

In this chapter, new techniques of enumeration will be developed based on the stratified representation of kinematic structures. Given a set of desirable kinematic properties, graphs are enumerated from abstract to concrete levels, keeping the increasing number of properties of graphs at each level as closely matched as possible to the given kinematic properties, thus minimizing the number of graphs (which corresponds to the amount of memory) and the number of eliminations (which contributes a great deal to computation time) during the entire enumeration process. The new method is selective because it will only enumerate graphs with some specified kinematic properties, and it is dynamic because it can be performed

3. Favorable Force Transmissions: The transmission angle between a dyad (i.e. a pair of binary links) should not deviate much from optimum. A poor transmission angle is unrealistic because real mechanisms would bind during operation.

Because algorithms are already available for sketching planar graphs without crossing edges, traditional approaches formulate the mechanism sketching problem as a problem of sketching a complex graph derived from the original graph of the mechanism. Adaptation of this approach to include more features, such as those listed above, makes it more complicated and computationally expensive, if not impossible.

The hierarchical approach, on the other hand, is straightforward conceptually and computationally. Fundamental properties which are common to a class of mechanisms are found from the graph at the most abstract level, which involves much fewer computations. More specific properties for the sketch are deduced from graphs at more concrete levels in the hierarchy. To add a new feature to a sketch, only procedures at the appropriate level of abstraction need to be modified.

Given the stratified code of a mechanism, the hierarchical sketching process can be described briefly as follow:

1. Contracted Level: Non-crossing. Divide major vertices into "peripheral" and "internal" ones based on the non-crossing criteria. All peripheral vertices will form the peripheral loop of the sketch, and all internal vertices should be located inside the peripheral loop.
2. Monochrome Level: Acceptable proportions. Determine the exact coordinates of each major vertices based on the number of binary vertices between each other. Since the coordinates of the vertices roughly indicate the location of links, this strategy will ensure a well proportioned sketch of graphs and mechanisms.
3. Colored Level: Favorable orientations. Sketch the mechanism based on the coordinates of the major vertices and the sequence of joint type of each string, which is directly available from the stratified code. Optimum transmission angles are ensured by adopting predetermined patterns to each different sequence of joint types.

The detailed process at each level, from abstract to concrete, is explained in the following sections. Examples are given in the last section, and sketches of 740 single degree-of-freedom kinematic chains are included in the Appendix along with the stratified codes.

7.6 CONCLUSION

The hierarchical approach to mechanism sketching divides the sketching problem into three stages:

1. Combinatorial: handled at the contracted level, ensures the non-crossing feature.
2. Geometrical: handled at the monochrome level, manages the approximate size of links.
3. Ornamental: handled at the colored level, manages proper orientations between binary links and other miscellaneous appearance of the sketch.

Each stage is handled at one level of abstraction to fulfill its most compelling requirement. This approach has three advantages:

1. Efficiency: A whole class of mechanism can be sketched by doing the combinatorial analysis (contracted level) or geometrical computation (monochrome level) only once. Noting that during a creative design process, sketching a class of mechanisms is usually required, the resulting effect can be enormous.

2. Flexibility: Different attributes (e.g. regularity, slenderness, etc.) can be modified to the whole sketch as well as to parts of the sketch. As a result, the designer has much more control over the appearance of the sketch.
3. Extendibility: Adding features and attributes (e.g. label ground link, impose a special geometry to a link for some special function, etc.) is much easier than it would have been by the traditional one-shot sketching process.

A comprehensive atlas of kinematic chains sorted by their stratified codes is included in the Appendix. Alternative vertex sequences which produce the same stratified code and traditional characteristic polynomials for monochrome graphs are included also for future reference. The sketches are immediate results of algorithms outlined in this chapter.

It has been demonstrated that the hierarchical approach is very effective in solving the problem of automatic mechanism sketching. Most problems in mechanism design involve considerations in combinatorial, geometrical and other aspects, hence a hierarchical methodology should benefit solutions to such problems as well.

Table V. Summary of Enumeration Results (Continued)

ENUMERATION OF SINGLE DEGREE-OF-FREEDOM PLANAR KINEMATIC CHAINS (Continued)												
No. of Inde- pen- dent Loops	Con- tracted			Bar- Linkage			Bi-Colored Theoretical			Practical R-P K.C.		
	G	#		G	#		G	#		G	#	
	R	of		R	of		R	of		R	of	
	A	V		A	V		A	V		A	V	
	P	V		P	V		P	V		P	V	
	H	S		H	S		H	S		H	S	
4	H		2	2	2.00		1044	1.23		88	1.30	
	J		6	3	1.67		10920	1.03		610	1.04	
	K		1	8	1.00		33024	1.00		1728	1.00	
	L		4	32	1.63		168912	1.06		4972	1.14	
	M		2	15	1.33		58496	1.00		1932	1.00	
	N		2	15	1.33		74360	1.02		2383	1.07	
	P		2	6	1.67		17888	1.05		520	1.11	
	Q		2	4	2.00		8384	1.11		186	1.21	
	R		8	42	1.29		296320	1.02		4508	1.06	
	S		2	43	1.16		272000	1.03		3978	1.07	
	T		1	10	1.00		56576	1.00		720	1.00	
	U		2	0	-		0	-		0	-	
	V		72	11	3.27		44424	1.13		315	1.63	
	W		12	28	1.79		165312	1.04		1113	1.19	
	X		4	8	2.25		31040	1.13		213	1.53	
	Y		4	3	3.33		5800	1.16		34	1.81	
	Z		6	0	-		0	-		0	-	
sum	17	7.76	230	1.54		1244500	1.04		23300	1.09		